VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

PROGRAM PRO SKRÝVÁNÍ DAT V OBRAZOVÝCH SOUBORECH

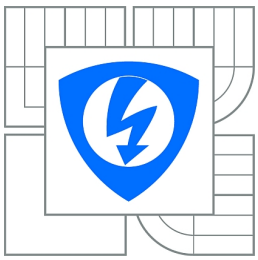
SOFTWARE FOR DATA HIDING IN IMAGE FILES

DIPLOMOVÁ PRÁCE
MASTER´S THESIS

AUTOR PRÁCE — Bc. Filip Morkus
AUTHOR

VEDOUCÍ PRÁCE — DOC. ING. KAREL BURDA, CSC.
SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Filip Morkus	***ID:***	88811
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Program pro skrývání dat v obrazových souborech

POKYNY PRO VYPRACOVÁNÍ:

Nastudujte a popište problematiku skrývání dat (steganografie). Na tomto základě navrhněte koncept řešení počítačového programu pro skrývání dat v obrazových souborech. Svůj návrh zdůvodněte a prakticky realizujte. Pro vytvořený program zpracujte manuál s praktickým příkladem použití programu. Při návrhu věnujte pozornost otázkám bezpečnosti dat a jednoduchosti použití programu.

DOPORUČENÁ LITERATURA:

[1] Žilka, R.: Steganografie a stegoanalýza. [Diplomová práce] Masarykova univerzita, Brno 2008.
[2] Krčmář, P.: Jak ukrýt tajná data do obrázku aneb steganografie v praxi. Root.cz, 30.4.2010.

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: doc. Ing. Karel Burda, CSc.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

ABSTRAKT

V této diplomové práci je postupováno od popisu základů a vzniku steganografie, jejích metod a účelu. Následuje detailní popis souborů typu BMP, do kterých bude prováděno skrývání dat. Struktura souboru je rozebrána do nejmenších detailů, aby nedošlo k poškození souboru při skrývání a zároveň aby bylo co nejoptimálněji využito informací uložených v hlavičce souboru. Dalším zaměřením bylo vysvětlení principu vlastní steganografické metody použité pro skrývání dat do souboru typu BMP.

Na základě těchto poznatků byl navržen koncept programu a následně vyhotoven daný program, který provádí samotné skrývání a odkrývání dat do obrazových souborů. K programu je vytvořen návod k použití a provedena demonstrativní ukázka funkce s popisem jednotlivých kroků.

V závěru práce je věnována pozornost vlivům, které mohou být pro danou metodu limitní a překročením těchto limitů by došlo k prolomení steganografické metody, což je prozrazení skrytých dat.

ABSTRACT

In my master's thesis I start from the description of the basics and the beginnings of steganography, its methods and purposes. Detailed description of BMP files with hidden data follows. A file structure is analysed into the smallest details to avoid damaging file while hiding. At the same time to make profit of the information included in a file header in the most effective way. Another aim was to explain a principal of steganographical method itself, used for hiding data into BMP file.

Based on these findings, a programme concept was designed and consequently that programme was made. It carries out the hiding and revealing data into the image files. Directions for use are made to this programme and there is also a demo of the features describing each step.

At the end of my master's thesis an attention is paid to those influences that can be limiting for this method. Exceeding these limits would lead to breaking the steganographical method, which means revealing the hidden data.

KLÍČOVÁ SLOVA

Steganografie, skrývání, odkrývání, program, obraz, soubor, data, rgb, bmp, bitmap

KEYWORDS

Steganography, hiding, revealing, program, image, data, rgb, bmp, bitmap

CITACE

MORKUS, F. Program pro skrývání dat v obrazových souborech. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 54 s. Vedoucí diplomové práce doc. Ing. Karel Burda, CSc..

PROHLÁŠENÍ

Prohlašuji, že svoji diplomovou práci na téma *Program pro skrývání dat v obrazových souborech* jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobních a jsem s plně vědom následků porušení ustanovení § 11 a následujícího autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne 26.5.2011

……

Bc. Filip Morkus

PODĚKOVÁNÍ

Děkuji vedoucímu mé diplomové práce Doc. Ing. Karlu Burdovi, CSc., předsedovi sekce – elektrotechnika, informatika, informační technologie, za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

Dále bych si dovolil poděkovat paní Ing. Miroslavě Hrouzové, techniko hospodářské pracovnici na děkanátu Fakulty elektrotechniky a komunikačních technologií, za velmi vstřícný a lidský přístup k řešení různorodých problémů v průběhu celého mého studia na VUT v Brně.

Rád bych zde také poděkoval svým rodičům a přítelkyni, kteří mě po celou dobu studia podporovali a vytvářeli vhodné zázemí pro plnění studijních povinností a tolerovali rozmary nálady způsobené tlakem končících různých termínů odevzdávání samostatných prací, zkouškovými obdobími, atd.

Zejména bych chtěl poděkovat a tuto práci pomyslně věnovat mému „nevlastnímu“ taťkovi panu Lubomíru Fuxovi, který se mnou a mými blízkými již bohužel případné promoce neoslaví v důsledku nedávné tragické nehody (+16.5.2011). Jeho výchově mimo jiné vděčím za to, jaký jsem a čeho jsem doposud dosáhl. Byl mi vždy maximální oporou, dobrým rádcem a velkým osobním vzorem.

DĚKUJI Lubí !!!

V Brně dne 26.5.2011

……

Bc. Filip Morkus

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

V této diplomové práci je popsána problematika steganografie, její historie, principy a možnosti. Mezi popsané možnosti patří například zvyšování bezpečnosti stenografických metod a to jak z hlediska modifikace datové části média použitého jako nosič, tak rovněž ke zvýšení bezpečnosti pomocí kombinace s metodami kryptografickými.

Následuje popis obrazových souborů, způsob ukládání dat, struktura uložených dat a objasnění použité steganografické metody na tomto typu media, včetně jejího vlivu na původní data a pozorovatelné změny při reprodukci. Z popsaných principů v těchto částech je navržen koncept programu, který slouží k ukrývání a též odkrývání dat ve formě libovolných souborů.

Vytvořený program podle navrženého konceptu je uživatelsky snadno použitelný. Pří tvorbě byl též kladen důraz na intuitivní ovládání. Přičemž poskytuje dostatek informací v průběhu užívání. K programu je vytvořen přehledný návod, ve kterém je provedeno demonstrativní skrytí a odkrytí dat s popisem jednotlivých kroků a jejich vlivu na výstupní část.

Část diplomové práce je zaměřena na poukázání dílčích faktorů ovlivňujících odolnost dané steganografické metody skrývání dat do obrazu. Dílčími faktory jsou například použitá modifikační hloubka, zdali je nosič barevný či černobílý, jaký druh obrazu nosič nese, atd.

1 STEGANOGRAFIE

Steganografie (z řeckých slov steganós – skrytý, gráphein - psát) je vědní obor zabývající se ukrýváním informace takovým způsobem, aby případný pozorovatel nebyl schopen tuto skrytou komunikaci vůbec zpozorovat. Její vznik není výdobytkem moderního věku, avšak sahá již do starověku, kde bylo již také zaznamenáno využívání steganografických metod. Příkladem může být ukrývání zprávy pod tenkou vrstvu vosku. K běžné komunikaci docházelo způsobem, že se na dřevěnou destičku nanesla tenká vrstva vosku, do kterého byla následně zpráva vyryta. Pod touto tenkou vrstvou byla však ukryta ještě zpráva vyrytá přímo do dřevěné destičky [1].

Steganografie je mnohdy mylně považována za podobor kryptografie [1]. Jde však o jiný obor. Ve většině steganografických postupů však dochází ke kombinacím s metodami kryptografickými.

Rozdíl mezi kryptografickými metodami a steganografickými metodami je zřejmý. Kryptografické metody aplikují na přenášenou informací různé algoritmy šifrování tak, aby informace byla pro nepovolaného pozorovatele nečitelná, ovšem tento pozorovatel si je vědom probíhající skryté komunikace. Zatímco steganografie aplikuje takové metody, aby nepovolaný pozorovatel probíhající skrytou komunikaci vůbec nezaznamenal [1].

1.1 PRINCIP STEGANOGRAFIE

Moderní steganografické metody jsou založeny na skrývání přenášených dat či zprávy v jiných datech kamuflující daný přenos. Data, ve kterých jsou přenášená data skrývána, se nazývají nosič. Steganogramem se rozumí výstup dané steganografické metody (obr. 1.1).

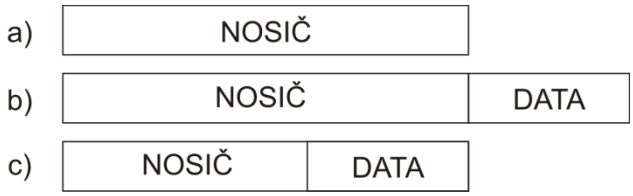
Obr. 1.1: Steganogram.

Je zřejmé, že při takovémto druhu skrývání informací musí být buď k nosiči přidávána redundantní data, nebo musí být data nosiče částečně pozměněna a to maximálně v takové míře, aby nedošlo ke znehodnocení informace přenášené nosičem anebo k již pozorovatelným změnám vedoucím k odhalení přenosu skryté informace (obr. 1.2), což by znamenalo prolomení použité steganografické metody.

Maximální množství těchto skrývaných dat se nazývá steganografickou kapacitou nosiče. Steganografická kapacita je pro různé typy nosičů rozdílná a je též závislá na zvolené míře redundance či maximální velikosti změny dat v nosiči.

Aby byla přenášená skrytá data hůře odhalitelná, nebo aby v případě odhalení přenosu skrytých dat nedošlo k přímému odhalení skrývané informace, kombinují se steganografické metody s metodami kryptografickými.

Při kombinaci těchto metod dojde nejprve k zašifrování přenášené informace některou z kryptografických metod a poté ke skrytí této zašifrované informace steganografickou metodou do nosiče.



Obr. 1.2: a) Steganografický nosič, b) redundantní skrytí, c) modifikace dat nosiče.

Jak již z předchozího textu vyplývá, pro skrývání přenášených dat se v dnešní době používají různé typy nosičů. Skrývání dat se provádí například na těchto nosičích:

- binárních souborech,
- zprávy síťových protokolů,
- souborových systémech,
- textu,
- obrazových souborech,
- audio souborech.

Všeobecně se dá říci, že vhodnými nosiči jsou datové segmenty, které mají proměnnou velikost, jejich vnitřní struktura je částečně modifikovatelná a jejich interpretace je záležitostí vnějšího subjektu.

Naopak nevhodnými či nepoužitelnými jsou nosiče s pevnou velikostí nebo nosiče bez vnitřní struktury (každý bit něco reprezentuje a modifikováním tohoto nosiče by došlo k znehodnocení jeho dat) [1].

1.2 STEGANOGRAFIE OBRAZOVÉHO NOSIČE

Stenografické metody lze aplikovat na veškeré typy obrazových souborů. Každý typ obrazového souboru má nějaké přednosti, ale též i úskalí. Hlavními hledisky pro výběr typu souboru, na kterém bude aplikována steganografie, jsou:

- rozšířenost typu souboru,
- vnitřní struktura,
- steganografická kapacita pro skrývání dat.

V této práci se zaměřím na steganografické metody skrývající informaci do bitmapových obrazových souborů. Důvodem je, že bitmapové obrazové soubory mají výborné předpoklady pro použití steganografických metod. Data obrazu jsou ukládána v „surové“ formě, tzn. bez komprese, mají tedy poměrně vysoké steganografické kapacity. Jelikož navrhovaný program bude ukrývat celé soubory, předpokladem je relativně velká datová velikost ukrývaných dat. V případě, že by měl být ukryt kupříkladu pouze text zprávy, nároky by byly nižší a vyhovující by byly například i obrazové soubory typu JPG či jiné komprimované typy souborů. Další předností bitmapových obrazových souborů je jejich jednoduchá datová struktura.

Skrývání informace do tohoto druhu nosiče je založena na nedokonalosti lidského oka, které není schopné do jisté míry rozpoznat změnu odstínu v barvě. Tato míra závisí jak na konkrétním jedinci, tak v nemalé míře na právě použitém

obrázku jako nosiče. V případě, že bude obrázek značně pestrý, například rozkvetlá louka, bude jeho steganografická kapacita větší než například u azurové oblohy. Tyto skutečnosti jsou lehce pozorovatelné na obr. 1.1, kde první obrázek je původní a na druhém je již provedeno ukrytí dat. V oblasti, kde je jehličí a dochází k častým změnám barev a odstínů, není poškození snadno pozorovatelné, kdežto v oblasti oblohy, kde jsou minimální rozdíly v barvě a odstínu, jsou pozorovatelné změny [1].

Jak je již patrné, při tomto druhu skrývání dat dochází k modifikaci dat nosiče. Datová velikost nosiče se tedy po použití metody nijak nezmění.

a)

b)

Obr. 1.3: Porovnání obrázků a) původní obraz, b) obraz se skrytými daty [1].

2 OBRAZOVÝ NOSIČ A JEHO DATOVÁ STRUKTURA

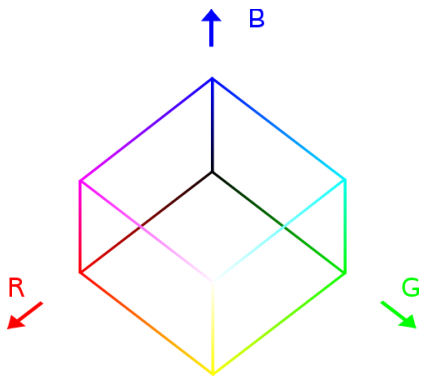
Jak již výše zmiňuji, v tomto návrhu budou preferovány bitmapové obrazové soubory. Nejrozšířenějším bitmapovým souborem je soubor BMP (BitMaP). V následujícím textu jsou podrobně popsány jeho vlastnosti.

2.1 STRUKTURA BITMAPOVÝCH OBRAZOVÝCH SOUBORŮ

Aby mohla být data skrývána v bitmapovém obrazovém souboru, je nutné znát jeho vnitřní strukturu.

Bitmapové obrazy jsou ukládány v RGB (Red, Green, Blue) modelu, každý pixel (obrazový bod) je reprezentován trojicí složek a to červenou, zelenou a modrou. Kombinací těchto tří složek je složena výsledná barva [2].

Na obr. 2.1 je znázorněna jednotková krychle RGB modelu, kde je vidět jakým způsobem jsou tyto barvy skládány.



Obr. 2.1: Jednotková krychle RGB modelu.

Ve většině případů se jednotlivé složky vzorkují 8 bity. Na obr. 2.2 je znázorněno bitové rozložení každé ze složek. Jeden pixel je tedy reprezentován 24 bity (tzv. TrueColor).

Obr. 2.2: Struktura RGB pro jeden pixel (TrueColor).

2.2 STEGANOGRAFIE NA RGB NOSIČI

Jak je již výše v textu zmíněno, v tomto případě se bude využívat steganografická metoda u které bude docházet k modifikaci stávajících dat nosiče, konkrétně dat RGB segmentu. RGB segmentem se rozumí trojice bajtů reprezentujících jednotlivé barvy jednoho pixelu, jak je znázorněno na obr. 2.2.

Steganografie se bude provádět přepisováním méně významných bitů skrývanými daty v jednotlivých složkách RGB. Steganografická kapacita bude tedy závislá na velikosti obrázku a počtu přepisovaných bitů v každé složce (obr. 2.3), je

dána vztahem 2.1. Je zřejmé, že čím více se bude navyšovat počet přepisovaných bitů, tím více bude růst rozdíl výsledného steganogramu od původní obálky a kvalita obrazu se bude zhoršovat. Není to dáno jen poměrem původních bitů vůči modifikovaným, ale též skutečností, že každý další modifikovaný bit ve složce má vyšší váhu. V tab. 2.1 je znázorněna závislost steganografické kapacity na velikosti obrazu a hloubce modifikace.

Obr. 2.3: Modifikace bitů podle použité hloubky.

$$c = 3 \cdot x \cdot y \cdot h \tag{2.1}$$

kde c [bit] – steganografická kapacita, x [px] – šířka obrazu, y [px] – výška obrazu a h [bit] – modifikační hloubka, 3 – konstanta udávající počet složek v jednom pixelu.

Tab. 2.1: Závislost steganografické kapacity na velikosti obrazu a hloubce modifikace.

Obraz			Pro hloubku modifikace		
šířka	výška	Mpix	3 bity/px	6 bitů/px	9 bitů/px
640	480	0,29	0,88 MBit	1,76 MBit	2,64 MBit
800	600	0,46	1,37 MBit	2,75 MBit	4,12 MBit
1024	768	0,75	2,25 MBit	4,50 MBit	6,75 MBit
1200	1024	1,17	3,52 MBit	7,03 MBit	10,55 MBit
1600	1200	1,83	5,49 MBit	10,99 MBit	16,48 MBit

2.3 OBRAZOVÉ SOUBORY TYPU BMP

Princip používané stenografické metody pro bitmapové obrazy byl tedy již představen. K návrhu stenografického programu je nutné znát též strukturu konkrétního souborového typu, který bude nosičem pro ukrývání dat. V tomto případě se bude jednat o soubory BMP.

Soubory BMP jsou jako většina datových souborů tvořeny dvěma základními částmi, a to částí záhlaví a vlastní datovou částí. Přičemž záhlaví se skládá z hlavičky a metainformací o uloženém datovém souboru [3].

2.3.1 Hlavička souboru BMP

Hlavička souboru typu BMP, je datovou strukturou BITMAPFILEHEADER, je dlouhá 14 bajtů a obsahuje informace o typu, velikosti a celkovém uspořádání dat v souboru. Vnitřní struktura je popsána v tab. 2.2 [3].

Tab. 2.2: Hlavička souboru BMP.

Název položky	Délka Položky	Význam
bfType	2 bajty	Identifikátor formátu BMP

bfSize	4 bajty	Celková velikost souboru s obrazovými údaji, některé aplikace tuto položku ignorují a dosazují nulu
bfReserved1	2 bajty	Tento údaj je rezervovaný pro pozdější použití, v současné verzi formátu BMP je zde vyžadována nula
bfReserved2	2 bajty	Tento údaj je rezervovaný pro pozdější použití, v současné verzi formátu BMP je zde vyžadována nula
bfOffBits	4 bajty	Posun struktury od BITMAPFILEHEADER od začátku vlastních obrazových dat

2.3.2 Metainformace souboru BMP

Tato část se nazývá BITMAPINFOHEADER a obsahuje základní metainformace o rastrovém obrazu, jako jsou rozměry, komprimační metoda a specifikace formátu rastrových dat. Délka této části je 40 bajtů. Datová struktura je popsána v tab. 2.3 [3].

Tab. 2.3: Metainformace souboru BMP.

Název položky	Délka položky	Význam
biSize	4 bajty	specifikuje celkovou velikost struktury BITMAPINFOHEADER
biWidth	4 bajty	Udává šířku v pixelech
biHeight	4 bajty	Udává výšku v pixelech
biPlanes	2 bajty	Počet bajtových rovin, v BMP je vždy rovna jedné
biBitCount	2 bajty	Počet bitů na pixel (1, 4, 8, 24)
biCompression	4 bajty	Typ komprimační metody (0 – BI_RGB, 1 – BI_RLE8, 2 – BI_RLE4)
biSizeImage	4 bajty	Velikost obrazu v bajtech, u nekomprimovaných obrazů může být zadána 0 (velikost se dá snadno vypočítat)
biXPelsPerMeter	4 bajty	Horizontální rozlišení v pixelech na metr, většinou se neudává a je nulová
biYPelsPerMeter	4 bajty	Vertikální rozlišení v pixelech na metr, většinou se neudává a je nulová
biClrUsed	4 bajty	Celkový počet barev, které jsou v použité v dané bitmapě. Jestliže je tato hodnota nastavena na nulu (což provádí většina aplikací), znamená to, že bitmapa používá maximální počet barev.
biClrImportant	4 bajty	Udává počet barev, které jsou důležité pro vykreslení bitmapy. Pokud je tato hodnota nulová (téměř vždy), jsou všechny barvy důležité.

2.3.3 Datová část souboru BMP

V datové části souboru BMP jsou již uvedena konkrétní obrazová data. Její velikost je přímo úměrná rozměrům obrazu.

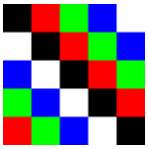
Data v této části nejsou ukládána pro každý pixel v posloupnosti RGB, ale posloupnost dat je BGR (Blue, Green, Red), což by mohlo být lehce matoucí. Schéma zápisu jednotlivých pixelů je znázorněno na obr. 2.4., kde $p_{x,y}$ je pixelem obrazu, přičemž index *x* označuje řádek a index *y* sloupec v obraze.

$p_{4,0}$	$p_{4,1}$	$p_{4,2}$	$p_{4,3}$	$p_{4,4}$
$p_{3,0}$	$p_{3,1}$	$p_{3,2}$	$p_{3,3}$	$p_{3,4}$
$p_{2,0}$	$p_{2,1}$	$p_{2,2}$	$p_{2,3}$	$p_{2,4}$
$p_{1,0}$	$p_{1,1}$	$p_{1,2}$	$p_{1,3}$	$p_{1,4}$
$p_{0,0}$	$p_{0,1}$	$p_{0,2}$	$p_{0,3}$	$p_{0,4}$

Obr. 2.4: Schéma zápisu BGR složek do datové části BMP souboru.

2.3.4 Analýza BMP souboru

Pro lepší orientaci a představení si výše popsané struktury souboru BMP jsem si vytvořil obrázek o velikosti 5x5 pixelů, na kterém se po jednom pixelu střídají barvy černá, červená, zelená, modrá a bílá, přičemž řádky jsou navzájem posunuty vždy o jeden pixel (obr 2.5).



Obr. 2.5: Analyzovaný BMP soubor (12x zvětšený).

Tab. 2.4: Hodnoty barev v DEC a HEX.

Barva	Hodnota dekadicky	Hodnota v hexa
Černá	0, 0, 0	00 00 00
Červená	0, 0, 255	00 00 FF
Zelená	0, 255, 0	00 FF 00
Modrá	255, 0, 0	FF 00 00
Bílá	255, 255, 255	FF FF FF

RGB data by měla být tedy dle tab. 2.4 a struktury RGB obrazu následující datovou posloupnost:

```
00 00 FF 00 FF 00 FF 00 00 FF FF FF 00 00 00
00 FF 00 FF 00 00 FF FF FF 00 00 00 00 00 FF
FF 00 00 FF FF FF 00 00 00 00 00 FF 00 FF 00
FF FF FF 00 00 00 00 00 FF 00 FF 00 FF 00 00
00 00 00 00 00 FF 00 FF 00 FF 00 00 FF FF FF
```

Na obr. 2.5 je zobrazená datová struktura analyzovaného obrázku. Ve vrchní části je hnědou barvou označena hlavička souboru, fialovou barvou metadata souboru a zbylá část jsou konkrétní data obrazu v BGR.

```
42 4D 86 00 00 00 00 00|00 00 36 00 00 00 28 00
00 00 05 00 00 00 05 00|00 00 01 00 18 00 00 00
00 00 50 00 00 00 00 00|00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00|FF 00 FF 00 FF 00 00 FF
FF FF 00 00 00 00 00 FF|00 FF 00 00 FF FF FF 00
00 00 00 00 FF 00 FF 00|FF FF FF 00 00 00 00 00
00 FF 00 FF 00 00 FF FF|FF 00 00 00 00 00 FF 00
FF 00 FF 00 00 00 00 00|00 00 00 FF 00 FF 00 FF
00 00 FF FF FF 00      |
```

Obr. 2.6: Vnitřní datová struktura BMP souboru.

Při porovnávání předpokládaných hodnot datové části jsou patrné nesrovnalosti. První nesrovnalostí je celkový počet bajtů, který by měl být 75 bajtů (5px × 5px × 3B), ale v souboru je 80 bajtů. Po hlubší analýze několika souborů a různém rozlišení jsem došel závěru, že součet bajtů jednoho řádku musí být dělitelný 4. Pokud tomu tak není, jsou na konci řádku doplněny nulové bajty (v obr. 2.6 zeleně označené) tak,

aby bylo této podmínce vyhověno. V případě analyzovaného obr. 2.5 jde o 1B na řádek (5px × 3B + 1 B = 16B).

2.4 SHRNUTÍ POPSANÝCH DŮLEŽITÝCH SKUTEČNOSTÍ

V tuto chvíli jsou již známé veškeré skutečnosti potřebné k detailnímu navržení steganografické metody pro ukrývání dat v nosiči jakým je soubor typu BMP. Detailně je popsán způsob jakým se data ukrývají a dále je detailně popsána datová struktura souboru BMP.

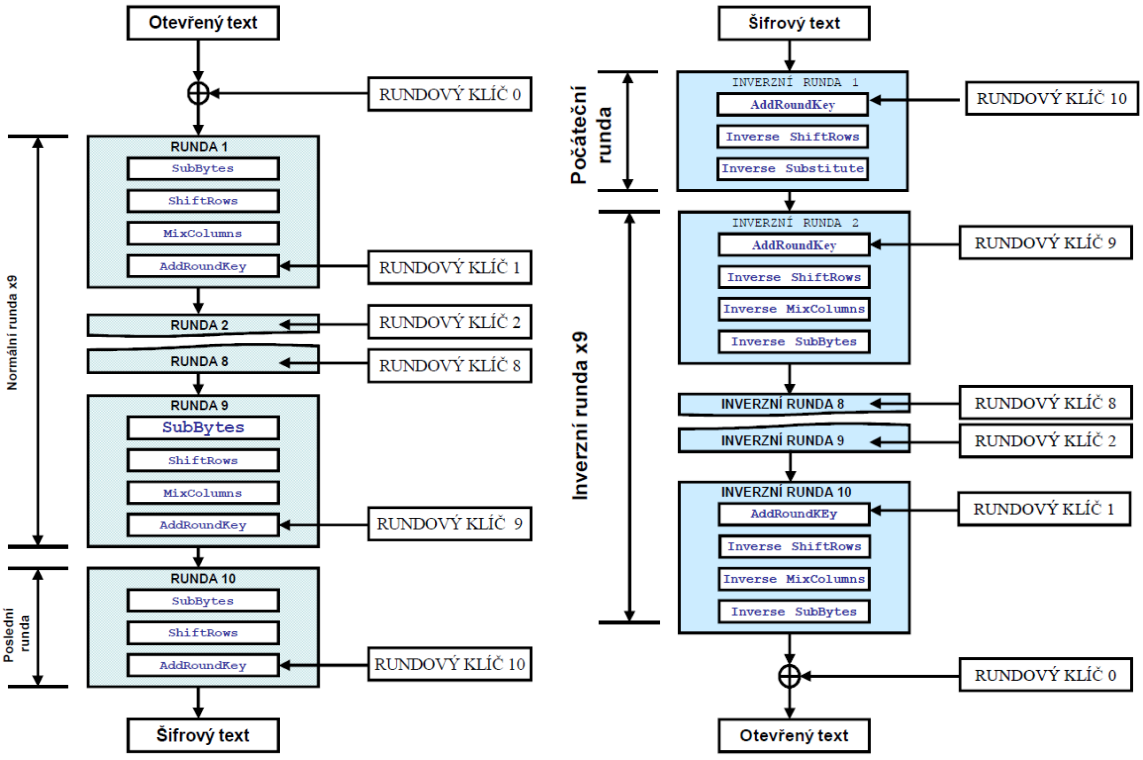
Znalost struktury je velice důležitá a to z titulu zamezení poškození nosiče při ukrývání dat a současně aby nedošlo k přepsání dat, které by mohly upozornit na skrytá data a tím umožnit prolomení stenografické metody. Proto části popisující datovou strukturu byla věnována relativně velká pozornost.

Znalostí hlavičkových informací a metainformací, které jsou v souboru uváděny (velikost obrazu, verze BMP, počet použitých bitů na jeden pixel,…) si lze i výrazným způsobem ulehčit práci při zjišťování informací o souboru potřebných pro zjištění steganografické kapacity a též při vlastním programování.

3 ŠIFROVÁNÍ

Jak bylo zmíněno v části 1.1, je vhodné při ukrývání dat tato ukrývaná data ještě zašifrovat některou ze šifrovacích metod. Optimálním řešením pro tento případ je šifrování blokovou šifrou AES. Volba této metody šifrování byla celkem jednoznačná, v současnosti se jedná o jednu z nejpoužívanějších metod pro šifrování. Její zabezpečení je na dostatečné úrovni a implementace je snadná díky dostupným knihovnám importovatelným do vlastního programu. V programu se tedy nebude celá funkce šifrování nikterak programovat, pouze se zavolá importovaná funkce, a to s patřičnými vstupními parametry, jejímž výstupem budou zašifrovaná data.

AES (Advanced Encryption Standard) je šifrovacím standardem schváleným NIST (National Institute od Standards and Technology). Jedná se o iterační blokovou šifru, což znamená, že zpracovává více znaků zdroje najednou. Délka šifrovaného bloku je 128 bitů. Délka klíče může být nezávisle volena ze 128, 192, 256 bitů [4]. Zjednodušený princip šifrování a dešifrování je znázorněn na obr. 3.1 [5]. Detailní popis šifrování v případě potřeby lze nalézt ve zdrojové literatuře [4][5].



Obr. 3.1: Princip AES a) šifrování, b) dešifrování.

SubBytes	-	bajtová substituce
ShiftRows	-	řádková rotace
MixColumns	-	matematické operace se sloupci
AddRoundKey	-	Zanesení klíčové závislosti v rundě

4 NÁVRH KONCEPTU PROGRAMU

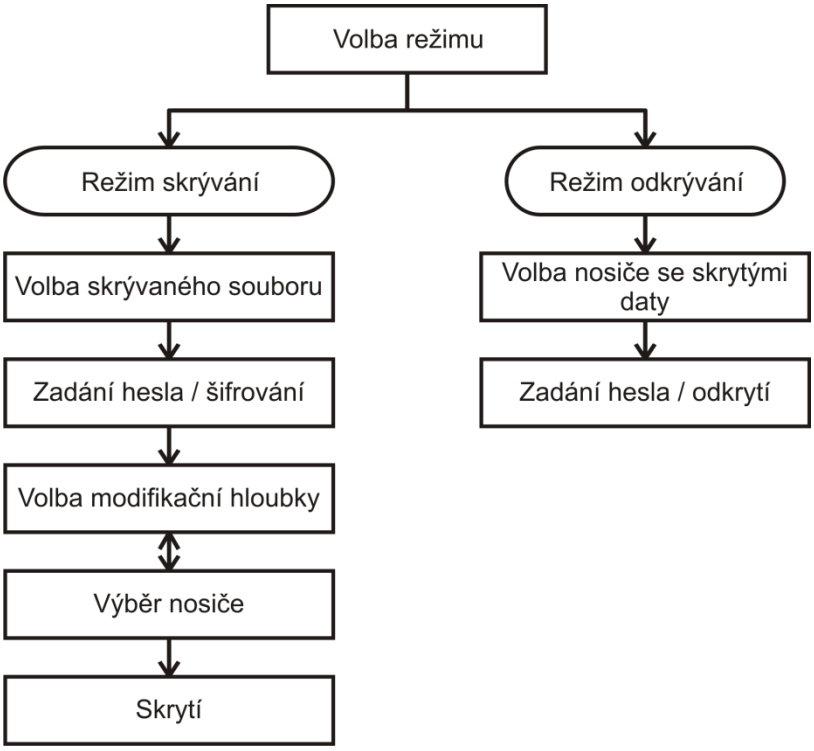
Nyní jsou již známy všechny potřebné skutečnosti, jejichž znalost je nutná k návrhu vlastního programu. Struktura nosiče, způsob jakým bude docházet ke skrývání dat i jak bude aplikováno šifrování.

V následujícím textu je popsán koncept programu, přičemž jednotlivé kapitoly představují dílčí části návrhu, ve kterých je probrána požadovaná funkce této dílčí části a případně dochází ke zdůvodnění učiněných rozhodnutí.

4.1 UŽIVATELSKÉ PROSTŘEDÍ

Z hlediska uživatele, bude obsluha programu navrhována tak, aby sám program uživatele provázel jednotlivými kroky a vždy bylo zřejmé, jaká činnost je po uživateli vyžadována. Nepůjde o konsolovou aplikaci, ale o aplikaci využívající dialogových oken.

Na obrázku 4.1 je naznačena topologie ovládání navrhovaného programu na úrovni uživatelského rozhraní. Jak je z obrázku zřejmé, jednotlivé úkony jdou postupně za sebou bez možnosti vrácení se o úroveň výše. Tato možnost je ponechána pouze v případě kroků, kde dochází k volbě modifikační hloubky a výběru nosiče. Důvodem ponechání této možnosti je možnost vzniku stavu, kdy nelze při zvolené modifikační hloubce data do nosiče ukrýt, protože má malou steganografickou kapacitu. Zvýšením modifikační hloubky se tato kapacita může upravit na vyhovující úroveň (vztah 2.1).



Obr. 4.1: Topologie ovládání navrhovaného programu.

Posloupnost úkonů vyžadovaných po uživateli v navrhovaném programu je tedy následující:

- Uživatel zvolí, zda bude data chtít skrývat či odkrývat

Skrývání dat

- Uživatel bude požádán o skrývaná data.
- Následuje požadavek o zadání hesla.
- V tomto okamžiku bude vyžadováno zadání počtu bitů přepisovaných v každém bajtu (hloubka modifikace).
- Dalším požadavkem bude výběr souboru typu BMP, který bude použit jako nosič.

Odkrývání dat

- Uživatel bude požádán o určení nosiče se skrývanými daty.
- Následuje požadavek na zadání hesla.
- Zobrazení oznámení o dokončení operace a případných doplňkových informací (umístění souboru, jeho název,…).

4.2 SKRÝVÁNÍ DAT - FUNKČNÍ POŽADAVKY

Posloupnost událostí v programu:

- Do programu se načtou data určená pro skrytí, obecně jakýkoli soubor.
- Skrývaná data se zašifrují.
- Načtení stupně modifikace – počet přepisovaných bitů na 1 bajt.
- Podle velikosti zašifrovaných dat a stupně modifikace se určí potřebné rozlišení v pixelech BMP souboru použitého jako nosič.
- Vyžádání nosiče - BMP souboru.
- Skrytí dat do nosiče.

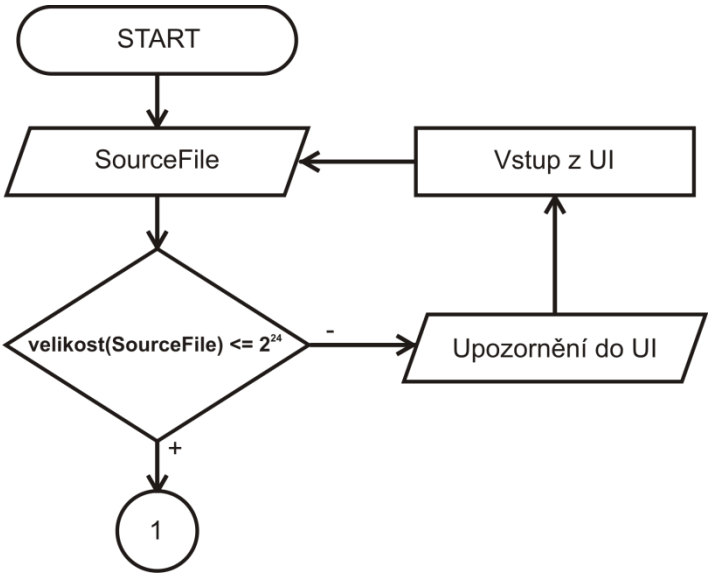
4.3 ALGORITMY JEDNOTLIVÝCH PROGRAMOVÝCH ČÁSTÍ PRO SKRÝVÁNÍ DAT

Podle jednotlivých bodů v 4.2 budou navrhovány algoritmy jednotlivých částí a popisována jejich funkce.

4.3.1 Načtení dat určených ke skrytí

V této části nebude docházet k žádným zásadním procedurám, půjde pouze o vstupní část programu. Je zde však vhodné místo, kde bude provedena korekce velikosti vstupního souboru. Korekce velikosti souboru bude prováděna ze dvou důvodů. Prvním důvodem je dostupnost nosičů s požadovanou steganografickou kapacitou. Vezmeme-li v úvahu, že by se uživatel programu mohl pokusit o skrytí například souboru o velikosti 100 MB a byla by použita hloubka modifikace 3 bity na 1 bajt, bylo by zapotřebí obrazu s rozlišením přesahující 93×10^6 px. Druhým důvodem je záznam délky skrývaných dat do pseudohlavičky (viz. 4.3.5) a rezervace místa v pseudohlavičce pro danou velikost skrývaných dat.

Zde bude tedy část, kde dojde k určení skrývaného souboru a jeho otevření pro čtení. Soubor nebude načítán do programu, ale půjde pouze o jeho určení. Samotné čtení dat souboru se bude provádět dle potřeby v části šifrování.



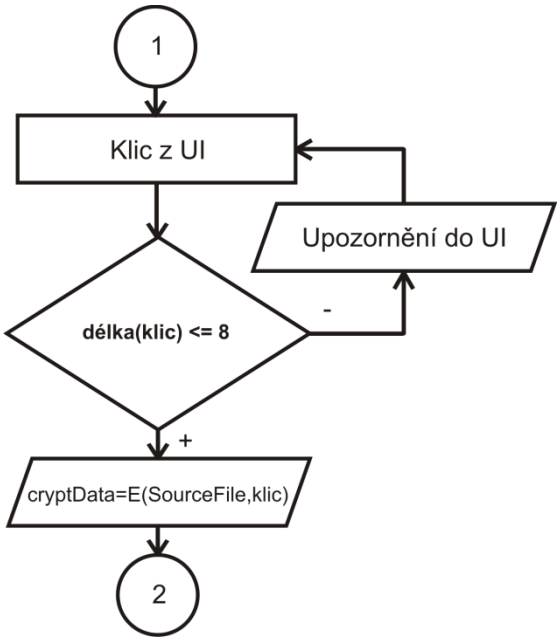
Obr. 4.2: Vývojový diagram: načtení dat ke skrytí.

4.3.2 Šifrování vstupních dat

Šifrování se provádí šifrou AES, jak je popsáno v části 3, jedná se o blokovou šifru. Data jsou zpracovávána po blocích, délka těchto bloků je 128 bitů.

Vlastním programováním šifrovací a dešifrovací metody se nebylo třeba zabývat. V programovacím jazyce Java, ve kterém je program programován jsou dostupné již vytvořené třídy/knihovny, které se snadno importují a pomocí jejich funkcí se bude šifrování a dešifrování provádět. Výstupem této části bude zašifrovaná datová struktura uložená v operační paměti počítače. Důvodem, proč se šifrovaná data nebudou rovnou ukrývat do obrazu, ale budou se ukládat do operační paměti počítače je skutečnost, že není ještě známa velikost dat pro skrytí a mohlo by dojít k situaci, kdy by se data nedala ukrýt, protože by nosič neměl dostatečnou kapacitu. Dalším důvodem tohoto zvoleného postupu je případná pozdější snazší modifikace programu.

V této části bude po uživateli požadováno zadání hesla pro provedení šifrování. U zadaného hesla bude požadována minimální délka 8 znaků, což je dnes minimálním standardem pro ochranu proti útoku hrubou silou.

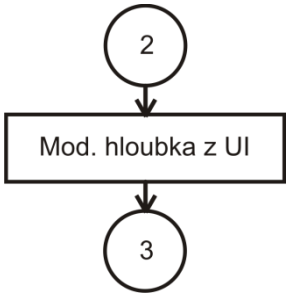


Obr. 4.3: Vývojový diagram: šifrování dat.

4.3.3 Určení stupně modifikace

V této části dochází ke zvolení parametru, který má výrazný vliv na steganografickou kapacitu, ale také na míru rozdílu vytvářeného a originálního obrazu.

Výběr se provádí z přednastavených hodnot *Auto, 1, 2, 3, 4, 5, 6, 7, 8* (čísla udávají počet modifikovaných bitů v jednom bajtu). Možnost výběru vyšších hodnot je spíše pro demonstraci možné míry poškození obálky než z praktického hlediska. Přičemž při výběru možnosti *Auto* dojde k automatickému výpočtu modifikační hloubky programem samotným a to takovým způsobem, aby došlo k co nejmenšímu poškození nosiče.



Obr. 4.4: Vývojový diagram: určení modifikační hloubky.

4.3.4 Výběr nosiče potažmo BMP souboru

V této části je uživatelem vybrán BMP soubor, který bude použitý jako nosič. Dochází zde také k výpočtu modifikační hloubky při volbě *Auto* a ověřování steganografické kapacity vybraného nosiče.

Po určení nosiče a jeho otevření dojde k ověření, zdali vybraný BMP soubor má dostatečnou steganografickou kapacitu pro zvolenou modifikační hloubku. Pro určení

je nutná znalost počtu pixelů obrazu. Počet pixelů BMP obrázku se snadno zjistí z metainformací vybraného souboru (viz 2.3.2), kde je uvedena šířka a výška obrazu v pixelech. Konkrétně jde o položky *biWidth* a *biHeight.* Steganografická kapacita v bajtech je odvozena ze vztahu (2.1) a je dána následujícím vztahem.

$$C = H \cdot x \cdot y \cdot \left(\frac{3}{8}\right)$$

(4.1)

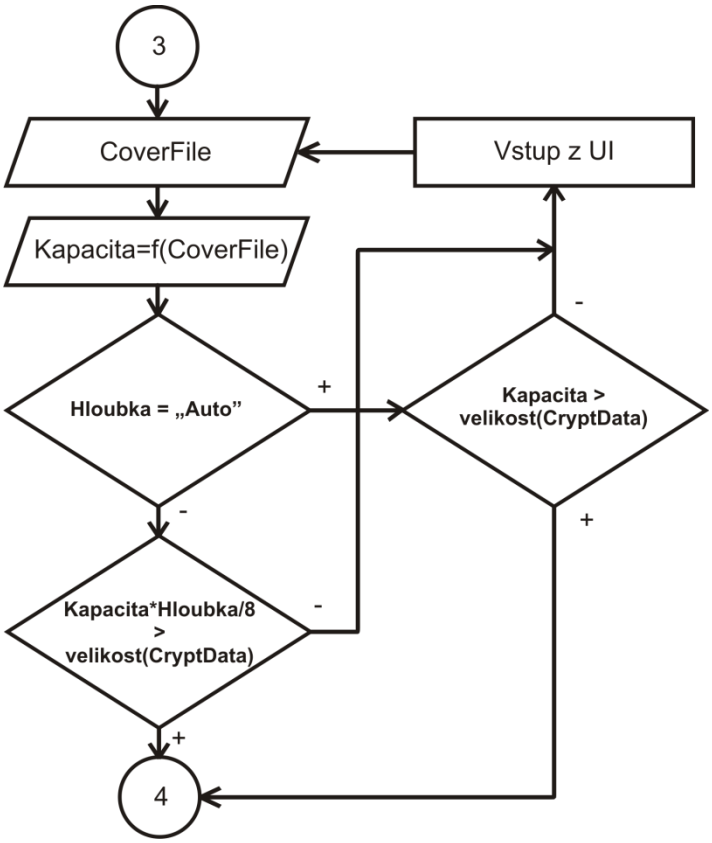
kde C [bajt] – steganografická kapacita, x [px] – šíře obrazu, y [px] – výška obrazu, H[bit] – modifikační hloubka, 3 – konstanta udávající počet složek a 8 – počet bitů v bajtu.

Pokud byla nastavena volba modifikační hloubky na *Auto*, je ve výpočtu steganografické kapacity uvažována maximální modifikační hloubka (8 bitů). V případě, že je vypočítaná steganografická kapacita větší jak velikost skrývaných dat, pokračuje se ve výpočtu optimální modifikační hloubky a to podle vztahu 4.3. Uživatel je o vyhovující či nevyhovující steganografické kapacitě stále informován a není mu umožněn další postup, dokud buďto nezvýší modifikační hloubku anebo neprovede změnu nosiče za nosič s větší steganografickou kapacitou.

$$H = \mathrm{RoundUp}\left(\frac{S}{3 \cdot N - D}\right)$$

(4.2)

kde H [b] – hloubka modifikace, S [b] – velikost zašifrovaného skrývaného souboru, N [px] – počet pixelů a D [b] – steganografická meta data.



Obr. 4.5: Vývojový diagram: výběr nosiče.

4.3.5 Skrytí dat do BMP souboru

Do této chvíle se dá říci, že šlo pouze o zadávání vstupních informací a teprve v tomto okamžiku bude prováděno vlastní skrývání dat (steganografie).

V první řadě musí být zjištěno, zdali v souboru BMP byly v řádcích doplňovány nějaké bajty (viz 2.3.4) a případně kolik. To proto, aby nedošlo k jejich přepsání. Přepsání těchto bajtů by sice nikterak daný nosič nepoškodilo, ani by v obraze zapsání těchto bajtů nebylo viditelné, ovšem využitím těchto doplňovaných nulových bajtů pro uložení skrývaných dat by značně oslabilo danou steganografickou metodu, protože by po analýze takového nosiče mohla být vlivem přepsání doplněných nulových bajtů odhalena steganografická metoda. Počet doplňovaných bajtů je dán vztahem 4.3.

$$\mathrm{N_{add}} = \mathrm{x\ MOD\ 4}$$

(4.3)

kde N_{add} [-] – počet doplňovaných bajtů, x [px] – šířka obrazu.

ZÁPIS DAT DO NOSIČE

Pro zajištění správného odkrývání skrývaných informací je potřeba zajistit, aby v nosiči byla kromě vlastních skrývaných dat, i data informující program při odkrývání o důležitých skutečnostech. Tato data se nazývají metainformacemi. Metainformace jsou uloženy v pseudohlavičce, jelikož soubor BMP použitý jako nosič již vlastní

hlavičku má a v ní nesmí dojít k žádné změně. Zaznamenávané informace budou také šifrovány. Jelikož data budou šifrována blokovou šifrou, šifrování bude probíhat po 128 bitových blocích. Po nastudování použitého šifrovacího standartu [6] byla jako optimální délka této pseudohlavičky zvolena délka 31 bajtů. Důvodem zvolení takovéto délky je skutečnost, že použitá šifrovací metoda popsána v [6] má již integrovanou funkci paddingu, kterým však v případě tohoto použití dochází k navýšení zašifrovaných dat o 1 bajt. Při této délce dochází k maximálnímu využití datového prostoru. Metainformace budou informovat o těchto vlastnostech:

- použitá modifikační hloubka,
- velikost skrývaných dat,
- původní název skrývaného souboru.

Jelikož je uvažována modifikace o hloubce 8 bitů, budou tato data zapsána celkem do 4 bitů. Odpovídající hodnoty jsou znázorněny v tab. 4.1.

Tab. 4.1: Vyjádření binárních hodnot odpovídajících modifikačnímu stupni.

Stupeň (Bit / Bajt)	Binární vyjádření
1	0001
2	0010
3	0011
4	0100
5	0101
6	0110
7	0111
8	1000

Velikost skrývaných dat bude reprezentována 24 bity, z čehož vyplývá, že maximální velikost skrývaných dat bude limitována na 16 MB, což je dáno vztahem 4.4. Tento limit je kompromisem mezi co největší kapacitou nosiče a dostupností tohoto nosiče. Pokud totiž budeme předpokládat maximální hloubku modifikace, což je 8 bitů/bajt odpovídá této velikosti obraz o rozlišení 5,6 Mpx. Tato velikost nosičů je ještě relativně snadno dostupná. K čemuž jsem došel statistickým pokusem, kdy jsem si v obrázkovém vyhledávači společnosti Google nechal nalézt obrázky typu BMP a zkoušel měnit minimální rozlišení hledaných obrázků. Do 6 Mpx je počet nalezených obrázků dostačující, překročením této velikosti je ovšem již nalezený výběr nízký.

$$2^{24} = 16.777.216\ \mathrm{B} \approx 16\ \mathrm{MB} \tag{4.4}$$

V tuto dobu je obsazených celkem 24 + 4 = 28 bitů z 248 bitů, zbývá tedy 220 bitů a není ještě proveden záznam původního jména souboru. Při záznamu znaků dochází k obsazení jednoho bajtu (8 bitů) jedním znakem. Zbylá kapacita tedy umožňuje zapsat 27 znaků.

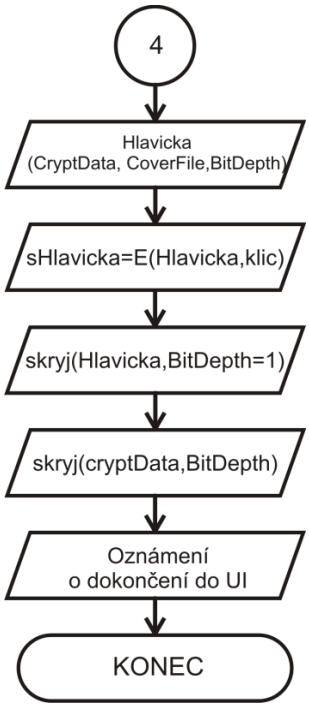
Pro lepší orientaci a chod programu bude minimální velikost jednotlivých položek v pseudohlavičce 1 bajt (8 bitů). Výsledná velikost položek je znázorněna na následujícím shrnutí datové náročnosti jednotlivých položek v pseudohlavičce kde je

v prvním sloupci minimální počet bitů pro záznam, v prostředním sloupci je počet bitů doplněných na bajty a v posledním sloupci je počet bajtů.

	Bitů	Bitů	Bajtů
Použitá modifikační hloubka	4	8	1
Velikost skrývaných dat	24	24	3
Původní název skrývaného souboru	220	216	27

Pseudohlavička je již zkonstruována a může se zaznamenat do nosiče. Záznam se v tomto případě bude provádět s implicitní modifikační hloubkou 1 bit/bajt. Je to nejmenší možná změna v nosiči, a to z důvodu minimálních změn v nosiči a současně proto, aby při odkrývání bylo zajištěno správné čtení dat zašifrovaných metainformací. Uživatelem zvolená modifikační hloubka tedy nemá žádný vliv na použitou hloubku při skrývání pseudohlavičky, ale až na skrývání zašifrovaných dat, které bude následovat.

Zápis skrývaných dat bude prováděn stejným způsobem jako zápis pseudohlavičky s tím rozdílem, že zde se bude již uplatňovat zvolená modifikační hloubka.



Obr. 4.6: Vývojový diagram: skrývání dat do nosiče.

4.4 ODKRÝVÁNÍ DAT – FUNKČNÍ POŽADAVKY

Posloupnost událostí v programu:

- otevření nosiče tj. souboru BMP,
- dešifrování pseudohlavičky,
- vyčítání zašifrovaných dat a ukládání dat do operační paměti,
- dešifrování uložených dat v operační paměti,

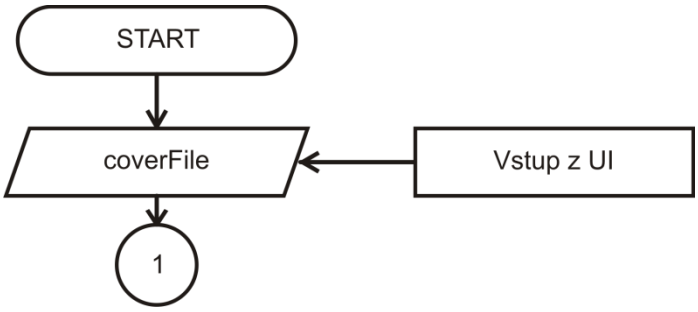
- vytvoření souboru s názvem uloženým v pseudohlavičce,
- uložení dešifrovaných dat do vytvořeného souboru.

4.5 ALGORITMY JEDNOTLIVÝCH PROGRAMOVÝCH ČÁSTÍ PRO ODKRÝVÁNÍ DAT

Tato část programu bude v podstatě funkčně inverzní oproti části pro skrývání dat, kde docházelo k importu skrývaných dat, nyní půjde o separaci. Bude však podstatně jednodušší a to z titulu absence konstrukce pseudohlavičky.

4.5.1 Otevření nosiče

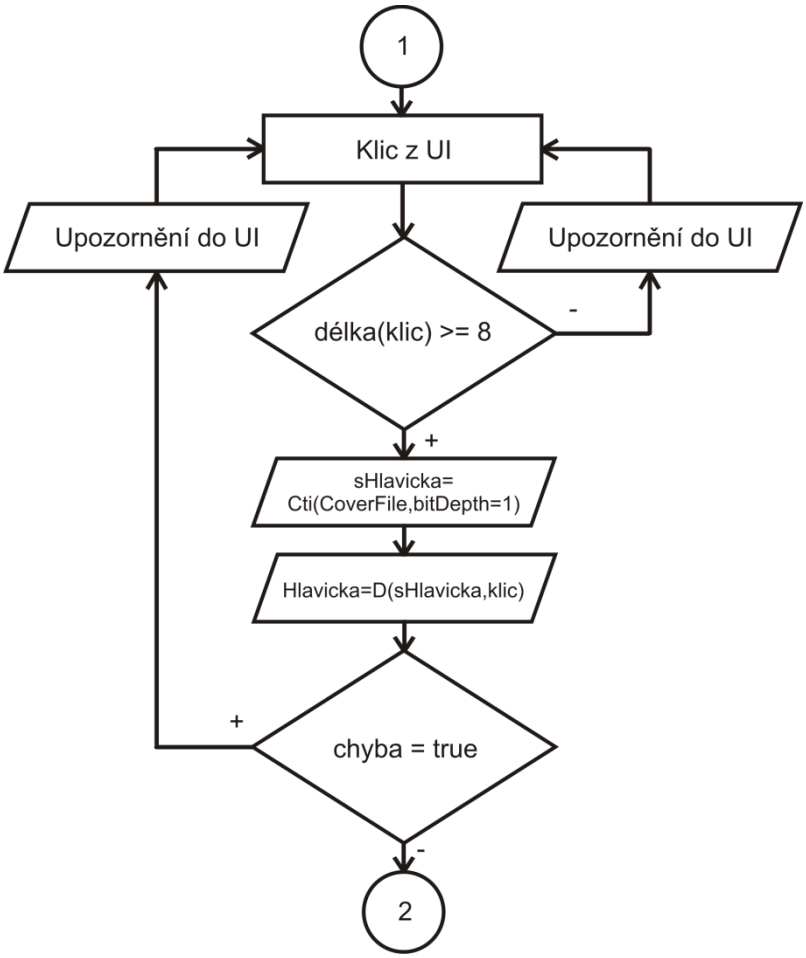
Otevřením nosiče se rozumí určení souboru typu BMP, ve kterém jsou skryta data a určena k jejich separaci. Nosič se nebude nikterak načítat, čtení vlastních dat se bude provádět až v dalších částech programu a to po blocích.



Obr. 4.7: Vývojový diagram: určení nosiče se skrytými daty.

4.5.2 Dešifrování pseudohlavičky

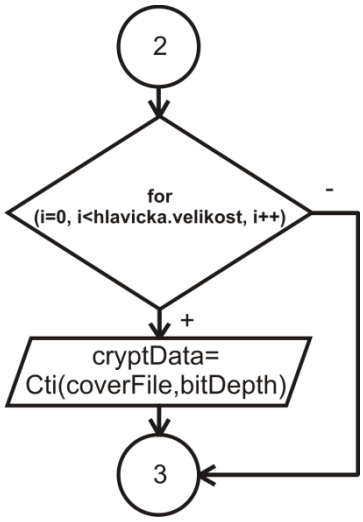
Podle klíče zadaného uživatelem se provede dešifrování pseudohlavičky, ve které jsou uložena důležitá data (jméno skrytého souboru, velikost skrývaného souboru a modifikační hloubka) pro další postup. Pseudohlavička je uložena v prvních 256 bitech (32 bajtů), což odpovídá necelým 11 px (11. pixel nese hlavičkové informace pouze ve dvou RGB složkách ze tří). V případě, že uživatel zadá chybné heslo, je tento stav detekován samotnou třídou, která v programu provádí dešifrování a pomocí použitého příznaku *chyba* je tento stav předán až do vrstvy uživatelského rozhraní, kde je uživateli oznámeno zadání chybného hesla.



Obr. 4.8: Vývojový diagram: čtení a dešifrování pseudohlavičky.

4.5.3 Vyčítání zašifrovaných dat a ukládání dat do operační paměti

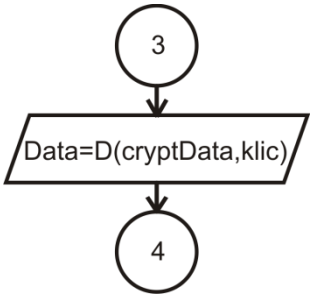
Tento úsek programu bude separovat uložená zašifrovaná data z obrazového nosiče a ukládat je do vytvořeného pole v operační paměti. Jaká data jsou v nosiči určena k separaci je dáno částí hlavičky, kde je uložena dálka skrývaných dat.



Obr. 4.9: Vývojový digram: separace zašifrovaných dat z nosiče.

4.5.4 Dešifrování uložených dat v operační paměti

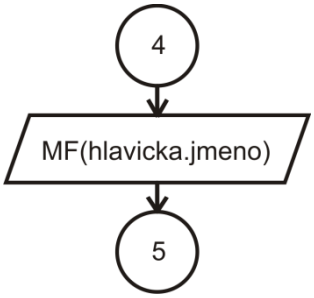
Po kompletní separaci zašifrovaných dat z nosiče se tato data dešifrují pomocí klíče zadaného uživatelem a uloží se opět do vytvořeného pole v operační paměti, kde budou připravena k zápisu do výstupního souboru.



Obr. 4.10: Vývojový diagram: dešifrování separovaných dat.

4.5.5 Vytvoření souboru s názvem uloženým v pseudohlavičce

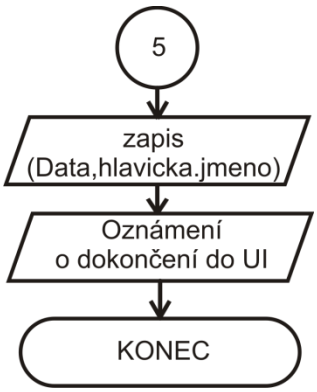
Tato část vytvoří a otevře k zápisu soubor s názvem, který byl uložen jako metainformace při skrývání dat. Jde již o výstup ze separace, do souboru se budou ukládat již koncová separovaná data.



Obr. 4.11: Vývojový digram: vytvoření souboru pro uložení odkrývaných dat.

4.5.6 Uložení dešifrovaných dat do vytvořeného souboru

Zde se dešifrovaná data v poli uloženém v operační paměti zapíší do dříve vytvořeného souboru a pouze se uživateli oznámí dokončení operace.



Obr. 4.12: Vývojový digram: zápis dat do výstupního souboru.

4.6 SHRNUTÍ ČÁSTI ZABÝVAJÍCÍ SE NÁVRHEM KONCEPTU PROGRAMU

V předchozí části jsem uvedl důležitá fakta, ze kterých jsem vycházel při samotném teoretickém konceptu vytvářeného programu. Tato koncepce byla v tuto chvíli pojata obecně. Termínem „obecně“ rozumím skutečnost, že nebyla zaměřena na nějaký konkrétní programovací jazyk. Proto se vlastní programová část sice bude držet dané osnovy, ale v detailech se může lehce odlišovat od zamýšlených postupů v této části navržených.

V konceptu programu jsem se zabýval korektním chodem programu a neuvažoval do detailu nekorektní vstupy. Nejedná se v tuto chvíli o opomenutí. Důvodem je, že každý programovací jazyk má jiné vlastnosti a jiné dovednosti tyto funkcionality zajišťující. Detailní postup bude popsán v následujících částech zabývajících se samotnou programovou částí.

5 REALIZACE PROGRAMU

V předchozí kapitole jsem nastínil koncept programu jako takový, ze kterého jsem vycházel při vlastním programování daného programu pro skrývání dat v obraze. Základem pro programování mi byly též poznatky i z předchozích kapitol, kde jsem detailně rozebíral strukturu BMP souborů a ostatních dílčích skutečností.

Jazyk, ve kterém jsem nakonec realizoval tvorbu vlastního programu, je Java [7]. K volbě tohoto programovacího jazyku mě vedla skutečnost, že jde o relativně perspektivní programovací jazyk s velkým potenciálem do budoucna. A rovněž jsem v tomto programovacím jazyku měl menší zkušenosti z předmětu LMUM (Multimédia). Další výhodou této platformy je relativní nezávislost na operačním systému, ve kterém je aplikace spouštěna.

Obr. 5.1: Logo platformy Java.

Vývojové prostředí, které jsem použil je volně dostupný program NetBeans IDE ve verzi 7.0 [8], se kterým jsem též měl již zkušenosti z předmětu LMUM. Toto vývojové prostředí má řadu výhod, přičemž těmi nejvýznamnějšími jsou velice uživatelsky přívětivá nabídka, kontextová menu, řada automatizovaných funkcí a intuitivní doplňování příkazů se zobrazováním základní nápovědy k danému příkazu.

Obr. 5.2: Informace o použitém vývojovém prostředí NetBeans IDE 7.0.

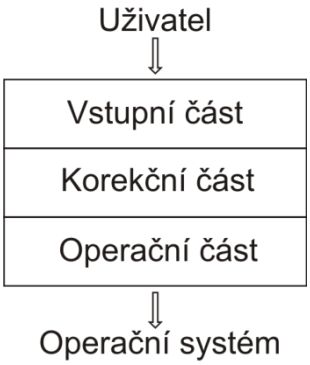
Vytvářený program jsem nazval *Data Camouflager*. Jak již sám název napovídá, jde o odvozeninu z anglických slov data a camouflage (maskování). Slovní spojení již samo o sobě směřuje k vlastnímu zaměření programu, což je skrývání (maskování) dat.

5.1 POPIS DÍLČÍCH KROKŮ V PROGRAMOVÁNÍ

Programování bylo rozděleno do několika dílčích kroků, respektive vývojových stupňů. Při programování byl též kladen důraz na průběžné vkládání komentářů k jednotlivým částem programu, což zvyšuje orientaci ve zdrojovém textu v případě pozdější modifikace i při vlastním programování.

V prvním kroku jsem vytvářel třídy, které byly zaměřeny k provádění určitých funkcí v samotném chodu programu. Byly pak snadno implementovatelné do konečného programu. Chod těchto tříd byl testován v jednoduché vytvořené konzolové aplikaci, kterou jsem si pro tuto potřebu vytvořil. V této konzolové aplikaci se vytvořené třídy použily k definování objektů, které si mezi sebou předávaly potřebné informace pro chod programu a případně průběžně vypisovaly mezivýsledky na obrazovku.

Po odladění jednotlivých tříd a ověření jejich správné funkce jsem začal s návrhem uživatelského rozhraní, ve kterém pak byly tyto třídy použity. V tomto rozhraní byla kromě vstupní části jednotlivých parametrů a vstupních souborů ošetřena i korektnost těchto vstupů. Korektnost vstupů ve vytvořených třídách nebyla ve většině případů ošetřena a byla přenechána až na tuto část. Dalo by se tedy říci, že vytvářený program má tři vrstvy a to vstupní, korekční, operační. Jak je naznačeno na obr. 5.3.



Obr. 5.3: Funkční vrstvy programu.

Vstupní část je samotný formulář, kde uživatel volí a zadává vstupní parametry pro požadované operace.

Korekční část kontroluje vstupní parametry, a pokud jsou mimo vymezené meze, jsou buď tato data upravena anebo je uživatel upozorněn na nekorektní vstupy.

Operační částí se rozumí samotné jádro programu, které provádí modifikaci dat, šifrování, čtení a zápis do souborů.

5.2 POPIS JEDNOTLIVÝCH TŘÍD

Celkem jsem si vytvořil 4 třídy, které se jmenují Cteni, Hlavicka, Aes a Kamuflaz. Jejich bližšímu popisu se budu věnovat v následujícím textu.

5.2.1 Třída - Cteni

V této třídě dochází k načtení souboru určeného ke skrývání a k šifrování načtených dat. Přičemž zašifrovaná data jsou uložena v operační paměti pro pozdější použití.

Globální proměnné a konstanty třídy:

Cteni.cryptData
- byte[]
- proměnná, ve které jsou uložená zašifrovaná data určená ke skrytí

Cteni.velikost
- int
- proměnná, ve které je uložena délka (velikost) skrývaných dat v bajtech

Cteni.sourceFile
- String
- celá cesta k souboru určeného ke skrytí

Procedury a funkce třídy:

Cteni.cestaZdroj();

inicializační procedura, ve které je zadána cesta k souboru, který je určený ke skrytí.

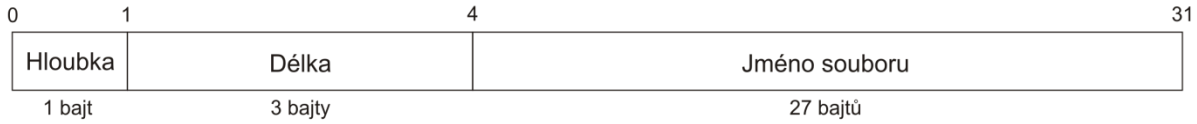
Cteni.Sifruj(String klic);

Procedura, která podle zadaného klíče zašifruje data z inicializovaného souboru a uloží je do proměnné cryptData.

5.2.2 Třída – Hlavicka

Třída Hlavicka obsahuje procedury a funkce určené ke konstrukci pseudohlavičky při skrývání dat. Ale též obsahuje procedury a funkce určené k vyčítání informací z této pseudohlavičky při odkrývání dat.

Délka pseudohlavičky je 31 bajtů a její struktura je naznačena na obrázku 5.4. Při šifrování nezáleží na délce šifrovaných dat, šifrovací metoda si sama jejich délku upraví patřičným způsobem (tzv. padding). Dojde však k navýšení výsledné délky zašifrovaných dat. V případě tohoto použití je to vždy maximálně o 1 bajt. Výsledná délka zašifrované pseudohlavičky je tedy 32 bajtů.



Obr. 5.4: Struktura pseudohlavičky.

Globální proměnné a konstanty třídy:

Hlavicka.chyba
- boolean
- proměnná, která slouží k přenosu chybných stavů při šifrování či dešifrování (např. nesprávné heslo)

Hlavicka.DELKA
- int
- konstanta určující délku nezašifrované hlavičky (v tomto případě 31 bajtů)

Hlavicka.hlavicka
- byte[DELKA]
- proměnná, ve které je uložena hlavičková struktura

Hlavicka.sHlavicka
- byte[]
- proměnná, do které jsou ukládána zašifrovaná hlavičková data

Procedury a funkce třídy:

Hlavicka.hloubka(int kolik);

Inicializační procedura pro konstrukci hlavičky. Slouží k zadání modifikační hloubky.

Hlavicka.velikost(long kolik);

Inicializační procedura, ve které je zadána velkost skrývaných dat v bajtech.

Hlavicka.jmeno(String text);

Inicializační procedura, ve které se ukládá původní název skrývaného souboru. V proceduře je ošetřena kontrola délky názvu s omezením na 27 znaků, přičemž se do délky započítává i tečka mezi jménem souboru a příponou. V případě, že je název delší dojde z levé strany k oříznutí. Dále zde také dochází ke korekci v případě použití diakritiky k nahrazení znaku s diakritikou znakem „~“. Toto opatření je v důsledku rezervované délky datového prostoru na jeden znak (8 bitů), což odpovídá znakové sadě ASCII v případě diakritiky je zapotřebí 2x větší prostor (16 bitů) pro uložení jednoho znaku, znaková sada Unicode.

Hlavicka.vypis();

Procedura sloužící k výpisu proměnné Hlavicka.hlavicka. Diagnostický nástroj pro ověřování správné konstrukce hlavičky.

Hlavicka.sufruj(String klic);

Procedura, která data uložená v Hlavicka.hlavicka zašifruje pomocí předloženého klíče a uloží je do Hlavicka.sHlavicka.

Hlavicka.Desifruj(String klic);

Procedura zajišťující dešifrování dat z proměnné Hlavicka.sHlavicka a uložení do Hlavicka.hlavicka. V proceduře je též ošetřeno předávání příznaku Hlavicka.chyba.

Hlavicka.getHloubka();

Výstupem této funkce je datový typ int udávající použitou modifikační hloubku uloženou v proměnné Hlavicka.hlavicka.

Hlavicka.getVelikost();

Výstupem této funkce je datový typ int udávající velikost skrývaných dat, která je uložená v proměnné Hlavicka.hlavicka.

Hlavicka.getJmeno();

Výstupem této funkce je datový typ String nesoucí jméno skrývaného souboru uložené v proměnné Hlavicka.hlavicka.

5.2.3 Třída – Aes

Tato třída zprostředkovává šifrovací a dešifrovací metody standartu AES. Šifrování a dešifrování je implementované do platformy Java již jako standart a to v knihovně javax.crypto.*.

Globální proměnné a konstanty třídy:

Aes.chyba
- boolean
- proměnná, která slouží k přenosu chybných stavů při šifrování či dešifrování (např. nesprávné heslo)

Procedury a funkce třídy:

Aes.sifruj(byte[] data, String klic);

Funkce datového typu byte[] na jejímž výstupu je pole zašifrovaných vstupních dat pomocí předloženého klíče.

Aes.desifruj(byte[] sData, String klic);

Funkce datového typu byte[] na jejímž výstupu je pole dešifrovaných vstupních zašifrovaných dat pomocí předloženého klíče. Ve funkci je zajištěno generování příznaku Aes.chyba v případě, že by došlo k chybě při dešifrování (např. v důsledku špatně zadaného hesla pro dešifrování).

5.2.4 Třída – Kamuflaz

Obsahem této třídy jsou již funkce a procedury zajišťující samotné skrývání a odkrývání dat do nebo z obrazového souboru BMP.

Globální proměnné a konstanty třídy:

Kamuflaz.bitDepth
- byte
- Proměnná, ve které je uložena hodnota určující hloubku bitové modifikace.

Kamuflaz.cryptData
- byte[]
- Proměnná, ve které je uloženo pole zašifrovaných dat určených ke skrytí.

Kamuflaz.coverFile
- String
- Proměnná udávající celou cestu k souboru, který je použit jako nosič.

Kamuflaz.finishFile
- String
- Proměnná uvádějící celou cestu k výstupnímu souboru.

Kamuflaz.jmenoSoub
- String
- Proměnná, ve které je uloženo jméno skrývaného souboru.

Kamuflaz.klic
- String
- Proměnná, v nichž je uložen klíč pro šifrování a dešifrování.

Kamuflaz.vel
- int
- Proměnná udávající velikost skrývaných dat.

Kamuflaz.chyba
- Proměnná, která slouží k přenosu chybných stavů při šifrování či dešifrování (např. nesprávné heslo).

Procedury a funkce třídy:

Kamuflaz.hlavicka(byte hloubka, int velikost, String jmeno, String klic);

Procedura sloužící k inicializaci hlavičkových parametrů určených ke skrytí.

Kamuflaz.kapacita(String bmpFile);

Funkce typu int která provádí výpočet steganografické kapacity souboru označeném v proměnné Kamuflaz.coverFile. Výstupem je maximální počet bajtů, které je daný obrazový soubor schopen skrýt.

Kamuflaz.zdroj(byte[] zdroj);

Procedura provádějící naplnění proměnné Kamuflaz.cryptData skrývanými daty.

Kamuflaz.cestaNosic(String nosic);

Inicializační procedura ukládající do proměnné Kamuflaz.coverFile cestu k souboru, jež je nosičem.

Kamuflaz.cestaVystup(String vystup);

Inicializační procedura ukládající do proměnné Kamuflaz.finishFile cestu k výstupnímu souboru.

Kamuflaz.kamufluj();

Procedura, ve které je již prováděna vlastní kamufláž dat (Kamuflaz.cryptData a Kamuflaz.hlavicka) do nosiče (Kamuflaz.coverFile) podle parametrů (Kamuflaz.bitDepth, Kamuflaz.finishFile), které byly zadány inicializačními procedurami.

Kamuflaz.odkryj();

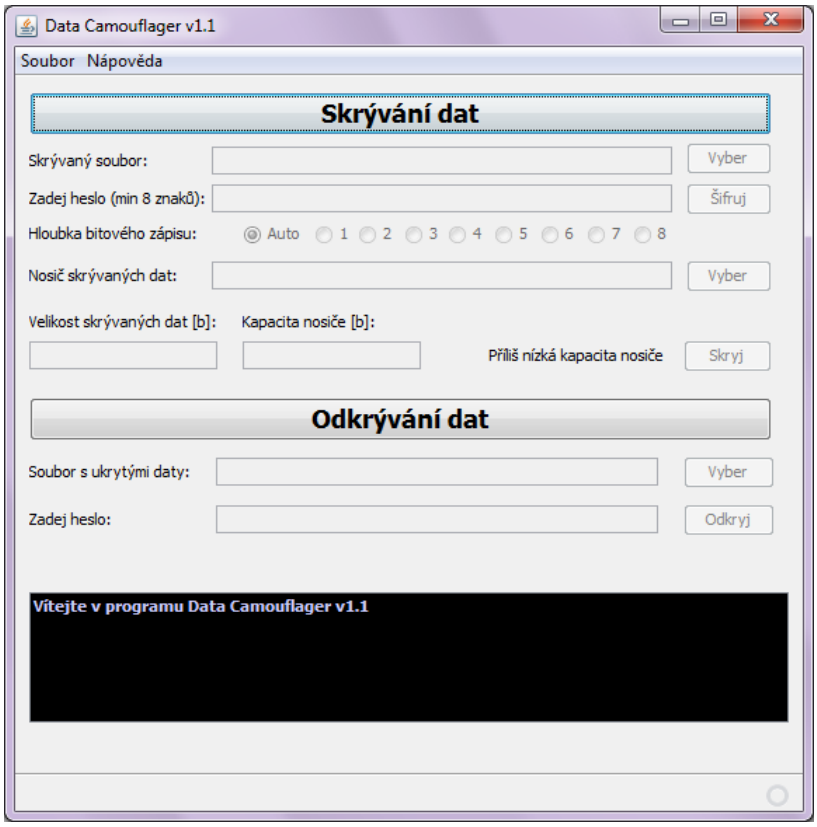
Procedura, která provádí se zde odkrývání dat z nosiče, jež je určen v proměnné (Kamuflaz.coverFile).

5.3 UŽIVATELSKÉ ROZHRANÍ

Uživatelské rozhraní jsem navrhoval podle kritérií popsaných v kapitole 4.1. Byl tedy kladen důraz na jednoduchost a přehlednost tohoto rozhraní. Ovládání je intuitivní a snadné. V rozhraní je vždy povoleno provádět pouze operace, které jsou v daném kroku vyžadovány. Zda je daná operace právě povolena či nikoli je ošetřeno formou aktivace či deaktivace jednotlivých ovládacích prvků.

Při zadávání vstupních parametrů je v případě potřeby zadaný parametr zkontrolován, a pokud je zadaný parametr nekorektní, je uživateli oznámena tato skutečnost v logovém okně. V tomto okně jsou též prezentovány důležité dílčí stavy chodu programu, jako je aktuálně prováděná operace, její dokončení a rekapitulace vstupních dat.

Na obrázku 5.5 je zobrazeno uživatelské rozhraní programu Data Camouflager ve výchozím stavu po spuštění. Na obrázku je patrné grafické rozdělení do tří částí. V první části (vrchní) jsou umístěny ovládací prvky pro skrývání dat. V druhé části (prostřední) jsou umístěny ovládací prvky pro odkrývání dat. A ve třetí části (spodní) je umístěno logové okno sloužící ke komunikaci s uživatelem.



Obr. 5.5: Uživatelské rozhraní programu Data Camouflager.

5.4 VLASTNOSTI PROGRAMU A SHRNUTÍ

Jelikož však hlavním zaměřením této diplomové práce není programování jako takové, ale spíše popis vlastností dané steganografické metody a prezentace jejích vlastností a aspektů, ve vytvořeném programu, není věnována detailní pozornost jednotlivým částem programu a jeho tvorbě. Pokud by však byla potřeba detailní znalost nějaké části programu, je ve zdrojovém kódu využíváno komentářů, kde se popisuje, jaké funkce daná část programu vykonává. Pro detailní znalost se dá tedy využít tohoto zdroje informací.

Program v aktuální podobě je uvažován spíše jako výukový, na kterém lze demonstrovat vlastnosti steganografické metody skrývání dat do obrazu typu BMP a různými konfiguracemi pozorovat vliv na výsledný nosič se skrytými daty. Ovšem jedná se o zcela funkční program, který základní dovednosti prezentuje v plné míře. V průběhu vývoje programu a při psaní samotné diplomové práce jsem postupně přicházel na různá zdokonalení programu, kterými jsou například:

- Využití hašovacích funkcí pro ověřování hesla. V současném programu je zadané heslo „násilně“ upraveno tak, aby vyhovovalo šifrovací a dešifrovací funkci. Heslo musí být 16 bajtů dlouhé, pokud tomu tak není a heslo je například příliš krátké, jsou zbývající bajty doplněny. V opačném případě, když je heslo příliš dlouhé, je od 17. bajtu řetězec znaků ignorován. V případě využití hašovací funkce by byl vstupní řetězec podroben hašovací funkci, která by měla definovanou délku výstupního haše, čímž by byla zaručena větší odolnost a jednodušší zpracování.

- Dalším zdokonalením by mohla být funkce zajišťující rovnoměrné rozprostření skrývaných dat do obrazu. V současné konfiguraci programu je sice možnost automatické volby modifikační hloubky, díky které je použita optimální modifikační hloubka, ovšem ani tato funkce nedovede zamezit vzniku hrany modifikované části obrazu a původní části obrazu. Díky čemuž se metoda při vyšších použitých modifikačních hloubkách stává slabší. Možným zamezením vzniku této hrany by byl výpočet poměru využitých a nevyužitých bajtů, respektive pixelů, před samotným skrýváním a nevyužité pixely nenechávat ve shluku na konec obrazu, ale rozptýlit je do celého obrazu tím, že by se ve vypočítaném poměru vždy určitý počet pixelů modifikoval. Pak by došlo k vynechání jednoho pixelu a opět by docházelo k modifikaci. Tento postup by se cyklicky opakoval až do vyčerpání zapisovaných dat.
- V aktuální podobě programu není ošetřen stav, kdy by se uživatel pokusil o odkrytí BMP souboru s neskrytými daty. V takovéto situaci by se program choval jak při korektním odkrývání, ale ve stádiu dešifrování by bylo uživateli oznámeno chybné zadání hesla. Tento stav by šel ošetřit například vložením identifikačních dat do pseudohlavičky.
- Jedním z komfortních vylepšení by bylo vložení tlačítka „Zruš“ do uživatelského rozhraní, které by provedlo vynulování všech dosud vložených parametrů. V aktuální podobě je jediným východiskem z této situace zavření a znovuotevření programu. Nejedná se sice o zásadní nedostatek, ovšem daná funkce by přispěla k vyššímu komfortu.
- Program neprovádí žádnou kontrolu verze BMP souboru a jeho barevnou hloubku. Mohla by nastat situace, kdy by obrazový soubor BMP nebyl s barevnou hloubkou 24 bitů anebo by byl uložen v jiné verzi, která by měla odlišnou datovou strukturu. V tomto okamžiku by došlo k poškození nosiče a obraz, který by nesl, by již nemusel být reprodukovatelný. Po dobu mého testování jsem se sice nesetkal s tímto problémem, ovšem reálné nebezpečí se zde nachází. Ošetření této situace lze provést pomocí ověření požadované hodnoty v hlavičkových údajích nesoucí informace o verzi a barevné hloubce.

Toto jsou návrhy pro zdokonalení další verze vytvořeného programu pro skrývání dat *Data Camouflager.*

6 NÁVOD K POUŽÍVÁNÍ PROGRAMU

V této kapitole popíši program *Data Camouflager* a provedu demonstrativní ukázku použití včetně popisu jednotlivých kroků.

6.1 POPIS PROGRAMU

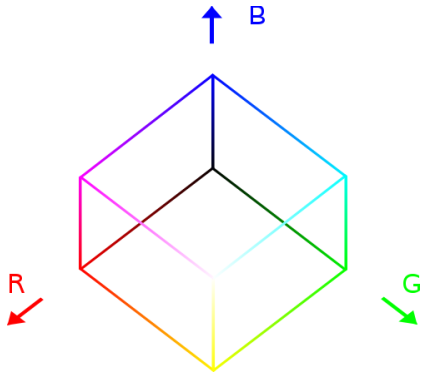
Program *Data Camouflager* je určen k provádění steganografické metody, ve které dochází ke skrývání dat do obrazu či k jejich zpětnému odkrývání. Přičemž v ideálním stavu není neznalý pozorovatel schopen identifikovat skrytá data v zobrazeném obrázku (což je principem steganografie) ani zobrazením konkrétních dat obrazového souboru.

Data Camouflager je tedy schopen skrýt jakýkoli soubor o maximální velikosti 16 MB do souboru typu BMP odpovídající velikosti, který je použit jako tzv. nosič.

Pro vysvětlení tohoto použití steganografické metody je zapotřebí znát jakým způsobem jsou ukládána obrazová data souborů BMP.

Bitmapové obrazy jsou ukládány v RGB (Red, Green, Blue) modelu, každý pixel (obrazový bod) je reprezentován trojicí složek a to červenou, zelenou a modrou. Kombinací těchto tří složek je složena výsledná barva.

Na obr. 6.1 je znázorněna jednotková krychle RGB modelu, kde je vidět jakým druhem jsou tyto barvy skládány.



Obr. 6.1: Jednotková krychle RGB modelu.

Ve většině případů se jednotlivé složky vzorkují 8 bity. Na obr. 6.2 je znázorněno bitové rozložení každé ze složek. Jeden pixel je tedy reprezentován 24 bity (tzv. TrueColor).

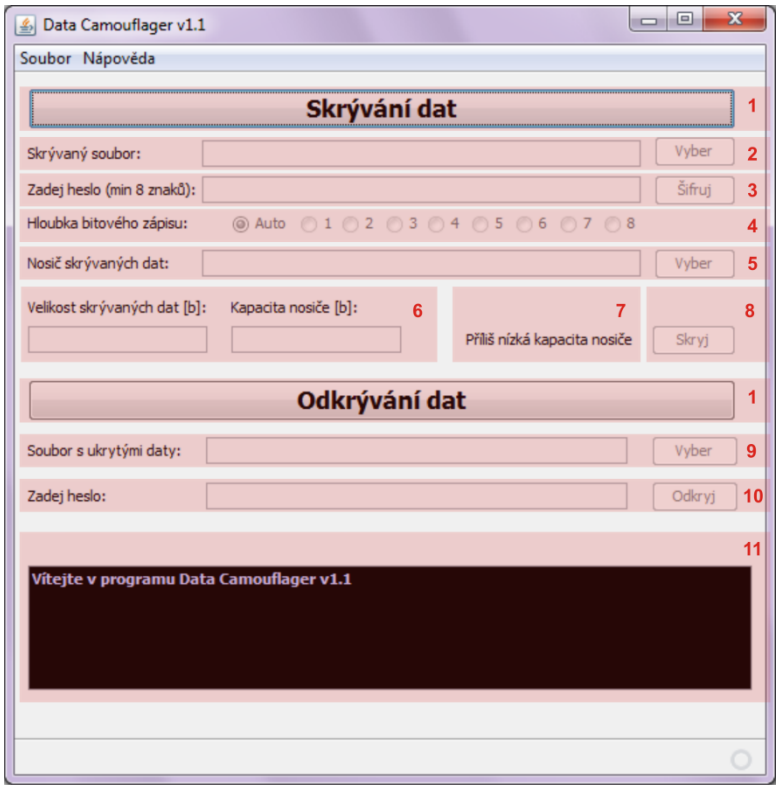
Obr. 6.2: Struktura RGB pro jeden pixel.

Vlastní steganografie se na RGB obrazech provádí přepisováním bitů jednotlivých složek od bitu s nejnižší váhou, kde je rozdíl původního pixelu od

modifikovaného nejmenší. Čím bude docházet k přepisu vyšších bitů, tím budou znatelnější změny ve výsledném obraze. A roste riziko odhalení použití steganografie na modifikovaném obraze. Ke znatelnějším změnám u modifikovaného obrazu vlivem rostoucího počtu použitých bitů pro skrytí dat není jen vlastní navýšení poměru použitých a původních pixelů, ale též skutečnost, že se zvyšujícím se počtem použitých pixelů roste i váha těchto bitů. K navyšování počtu použitých bitů k ukrytí dat, dochází za účelem zvýšení steganografické kapacity nosiče.

6.2 POPIS UŽIVATELSKÉHO ROZHRANÍ

Na obrázku 6.3 je vyobrazeno uživatelské rozhraní, které je barevně rozděleno do očíslovaných polí, přičemž jednotlivá pole budou níže popsána.



Obr. 6.3: Popis uživatelského rozhraní.

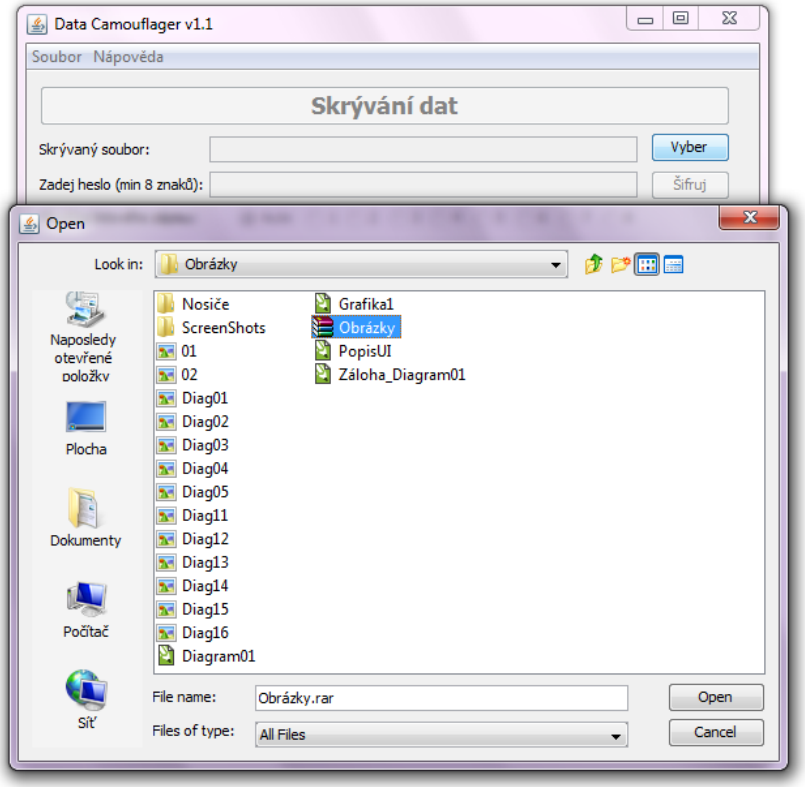
1 Tlačítka volby režimu skrývání či odkrývání dat.
2 Výběr souboru, který je určen ke skrytí (jakýkoli soubor do velikosti 16 MB).
3 Textové pole pro zadání hesla k následnému šifrování pomocí tlačítka „Šifruj“.
4 Výběr hloubky modifikace, případně automatického režimu zvolení této hloubky.
5 Výběr souboru typu BMP použitého jako nosič.
6 Informace o velikosti zašifrovaných skrývaných dat a kapacitě zvoleného nosiče při zvolené modifikační hloubce.
7 Informace, zdali je vybraný nosič a zvolená modifikační hloubka vyhovující pro následné skrývání dat do nosiče.
8 Tlačítko pro provedení zahájení operace samotného skrývání dat do nosiče.
9 Výběr souboru BMP použitého jako nosič se skrytými daty.

10 Textové pole pro zadání hesla k následnému dešifrování a tlačítko spouštějící odkrývání a následné dešifrování dat.
11 Logové okno ve kterém jsou průběžně vypisovány informační zprávy a chybová hlášení.

6.3 POUŽITÍ PROGRAMU DATA CAMOUFLAGER

6.3.1 Skrývání dat

Volbu režimu skrývání dat provedeme stisknutím tlačítka „*Skrývání dat*“, čímž dojde k aktivaci tlačítka „*Výběr*“ pro zadání cesty ke skrývanému souboru. Následuje tedy volba skrývaného souboru, po kliknutí na tlačítko výběr se otevře dialogové okno umožňující výběr jakéhokoliv souboru (obr. 6.4). V tomto případě došlo k výběru souboru „*Obrázky.rar*“ o velikosti 1,45 MB. Po potvrzení výběru se v logovém okně vypíše hlášení „*Data určena ke skrytí jsou načtena*“ a dojde k aktivaci pole pro zadání hesla a tlačítka šifruj.



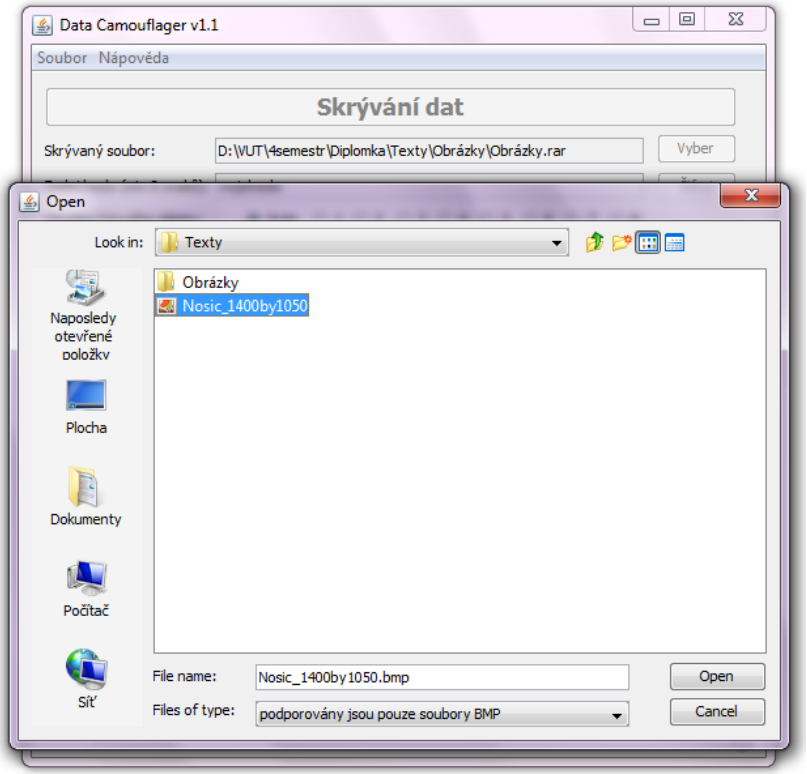
Obr. 6.4: Dialogové okno pro výběr skrývaného souboru.

Následuje zadání hesla o minimální délce 8 znaků, které je použito pro následné šifrování skrývaného souboru. Po stisknutí tlačítka je v logovém okně opět vypsáno upozornění o probíhajícím šifrování a jeho dokončení, případně o zadání příliš krátkého hesla.

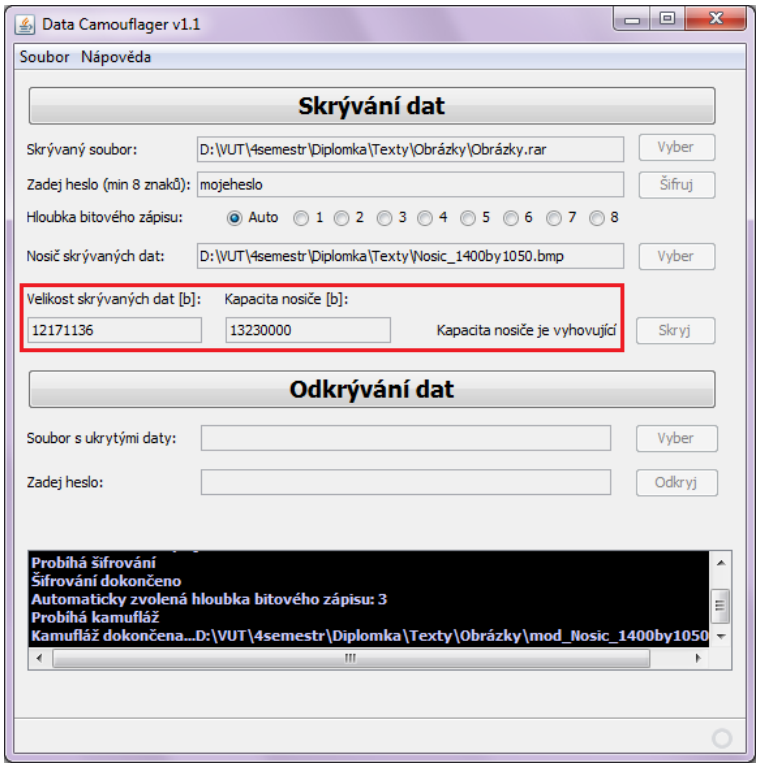
Dokončením šifrování jsou aktivovány výběrová tlačítka pro zvolení modifikační hloubky a tlačítko pro výběr BMP souboru použitého jako nosiče. Modifikační hloubkou se rozumí počet přepisovaných bitů v jednom bajtu. Možnosti jsou 1 – 8,

případně volba automatického zvolení modifikační hloubky. Při volbě automatického zvolení modifikační hloubky, dojde po výběru obrazového souboru BMP, který bude použit jako nosič k výpočtu optimální hloubky modifikace. Termínem optimální se rozumí stav, kdy má díky co nejmenší zvolené modifikační hloubce nosič dostatečnou steganografickou kapacitu pro skrytí šifrovaných dat. V tomto případě ponechám volbu modifikační hloubky na programu (režim „*Auto*“), který jí vypočítá z výše uvedených faktů.

Po výběru modifikační hloubky kliknutím na tlačítko „*Výběr*“ v řádku „*Nosič skrývaných dat*“ dojde opět k otevření výběrového dialogového okna, kde dochází již k filtraci typu souborů a jsou zobrazovány pouze soubory typu BMP (obr. 6.5). Pro tento příklad došlo k výběru souboru „*Nosic_1400by1050.bmp*“, který má rozlišení 1400x1050 pixelů a velikost souboru je 4,20 MB. Po potvrzení výběru je v tomto případě v logovém okně poskytnuta informace o automaticky zvolené modifikační hloubce, v daném případě jde o 3 bity/bajt. Dále jsou uživateli již dostupné kompletní informace o velikosti skrývaných zašifrovaných dat (nejsou totožná s velikostí skrývaného souboru) a velikosti kapacity při zvolené hloubce modifikace 6 bitů/bajt. Další informací je, zdali je nosič vyhovující či nikoli (obr. 6.6 - červené orámování). V tomto okamžiku je stále povolena možnost volby hloubky modifikace, díky čemuž se může demonstrovat vliv závislosti použité modifikační hloubky daného nosiče na výsledné steganografické kapacitě nosiče.



Obr. 6.5: Dialogové okno pro výběr souboru typu BMP použitého jako nosič.

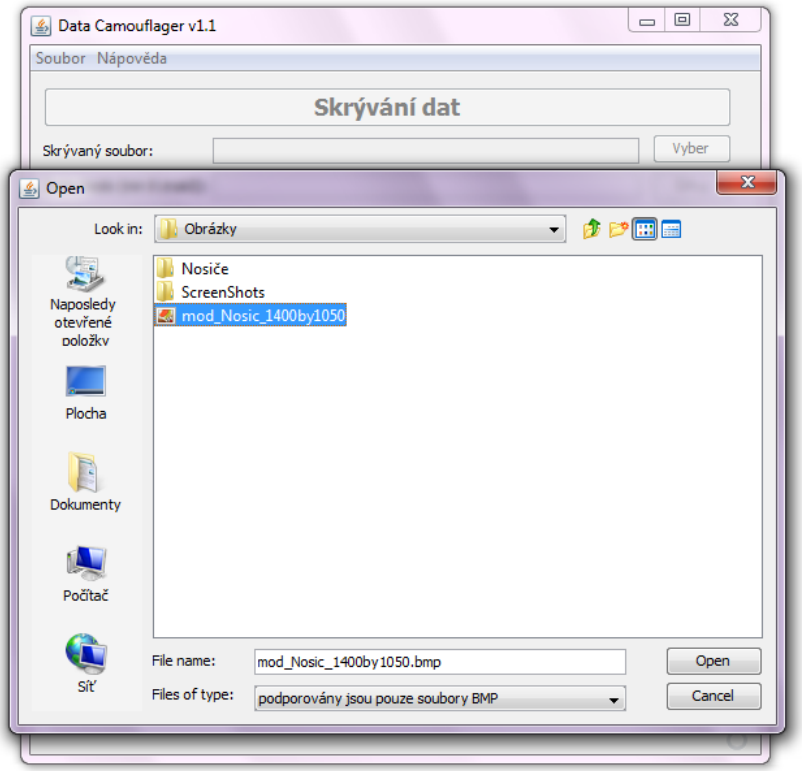


Obr. 6.6: Uživatelské rozhraní s kompletně zadanými parametry a po provedeném skrývání.

Po výběru vyhovujícího nosiče již zbývá pouze provést vlastní skrývání zašifrovaných dat do nosiče. Skrývání se provede tlačítkem „Skryj“. V logovém okně je opět uvedena informace o prováděné operaci „*Probíhá kamufláž*“, přičemž po dokončení této operace je uživateli oznámeno její dokončení a cesta k vytvořenému modifikovanému nosiči. Doba skrývání dat může v závislosti na velikosti skrývaných dat trvat relativně dlouhou dobu a to až v řádech jednotek minut. Cesta k nosiči je totožná s cestou skrývaného souboru. Jak je z posledního řádku v logovém okně (obr. 6.6) patrné vytvořený BMP soubor nese původní název nosiče ke kterému je na začátek připsán řetězec „*mod_*“. K připsání řetězce dochází z důvodu ochrany proti přepsání původního souboru v případě, že budou skrývaná data a nosič ve stejném adresáři. Po dokončení operace jsou aktivní opět pouze tlačítka pro volbu režimu skrývání či odkrývání dat.

6.3.2 Odkrývání skrytých dat

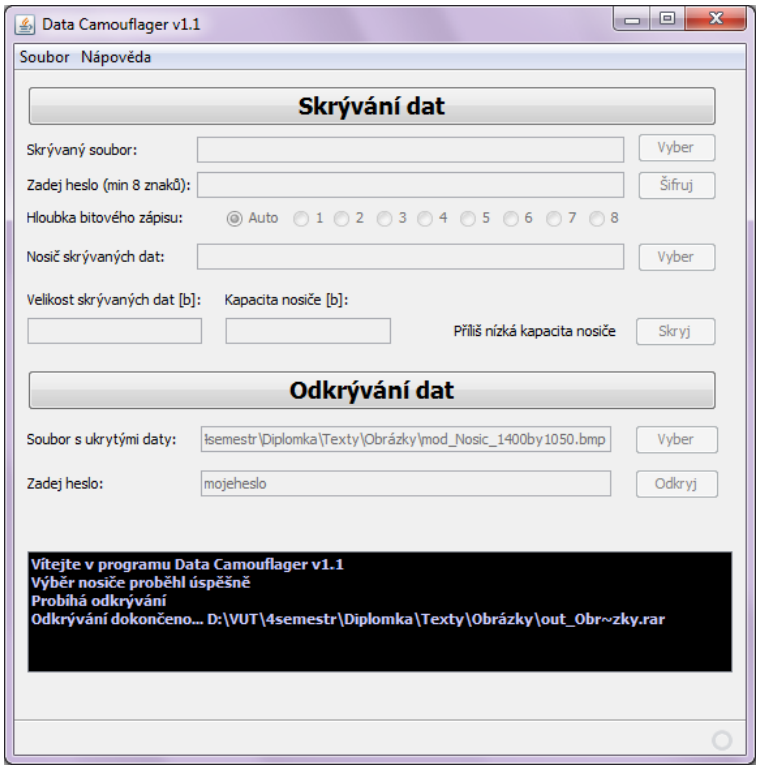
Pro volbu režimu odkrývání dat se z aktivních tlačítek zvolí tlačítko „Odkrývání dat“ čímž dojde k zaktivnění prvků pro odkrývání dat. První položkou pro uskutečnění odkrytí dat je výběrové tlačítko „*Vyber*“. Po jeho zvolení dojde k otevření výběrového dialogového okna, u kterého je opět aplikován filtr zobrazující pouze soubory typu BMP (obr. 6.7). V tomto případě došlo k výběru v souboru, který byl vytvořen v předchozí podkapitole a to souboru „*mod_Nosic_1400by1050.bmp*“. Provedení výběru je potvrzeno v logovém okně textem „*Výběr nosiče proběhl úspěšně*“. Dále dojde k aktivaci pole pro zadání hesla, pomocí kterého bude provedeno dešifrování a aktivaci tlačítka „*Odkryj*“.



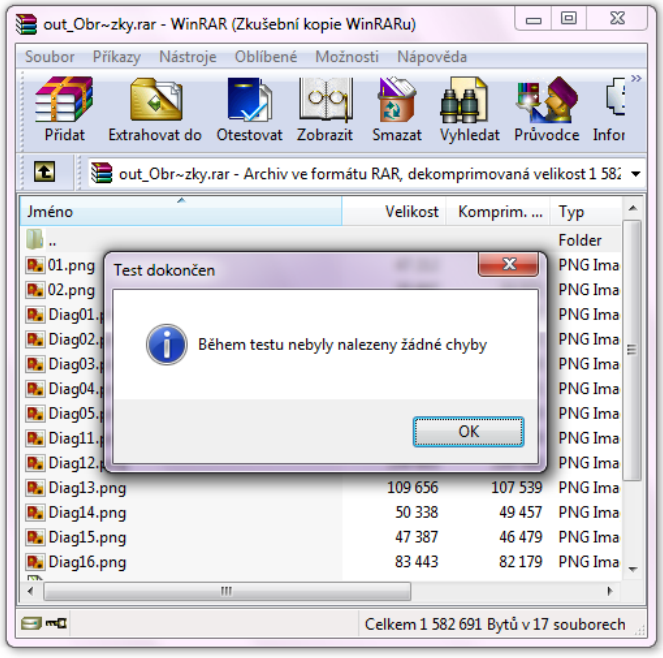
Obr. 6.7: Dialogové okno pro výběr nosiče se skrytými daty.

Do dialogového okna v řádku „*Zadej heslo*“ se vypíše stejné heslo, jako bylo použité pro skrývání souboru. Pokud dojde k zadání nesprávného hesla je to rovněž uživateli oznámeno v logovém okně a nesprávně zadané heslo může opravit do správného tvaru. V tuto chvíli je již vše potřebné pro odkrytí dat zadáno a samotné odkrývání se provede pomocí tlačítka „*Odkryj*“. Operace odkrývání může taktéž trvat relativně dlouhou dobu a to až v řádu jednotek minut. Počátek odkrývání je také oznámen v logovém okně a dokončení operace taktéž (obr. 6.8). Na obrázku je též vidět, že u oznámení dokončení operace je uvedena cesta k odkrytému souboru. Cesta je shodná s umístěním nosiče. Před jméno souboru je opět umístěn řetězec („*out_*“), který zamezuje přepsání případně stejnojmenného souboru. V praktickém použití by k přepsání v zásadě docházet nemělo, ale při zde uvažovaném výukovém využití je tato možnost přepsání pravděpodobnější. V posledním řádku souboru je také patrná změna názvu původního souboru z „*Obrázky.rar*“ na „*Obr~zky.rar*“. Tento jev je důsledkem limitované podpory znakové sady a znaky které nejsou obsaženy v ASCII sadě, jsou nahrazeny znakem „~“. Po dokončení operace odkrývání jsou opět aktivní pouze ovládací prvky pro volbu režimu skrývání dat do obrazu, nebo odkrývání dat z obrazu.

Jako skrývaný soubor byl záměrně vybrán soubor archívu typu RAR. Na souborech archívu, lze snadno ověřit, zda byla nějakým způsobem poškozena data skrývaného souboru, protože je téměř pravidlem, že archívy mají integrovánu kontrolu archivovaných dat, většinou CRC metodou. Pokud tedy odkrytý soubor „*out_Obr~zky.rar*“ půjde bez sebe menších problému dekomprimovat, je vše v naprostém pořádku. Test byl proveden v programu WinRAR x64 verze 3.92. Výsledek testu je zobrazen na obrázku 6.9.



Obr. 6.8: Uživatelské rozhraní s kompletně zadanými parametry a po provedeném odkrývání.



Obr. 6.9: Kontrola zdali nedošlo k poškození dat skrýváním či odkrýváním souboru „*Obrázky.rar*“.

6.3.3 Souhrn kapitoly

V této kapitole byl detailně popsán princip použité steganografie, program samotný a postup skrývání i odkrývání dat. V popsaném postupu je snadno viditelné, že použití programu Data Camouflager je jednoduché a jeho ovládání je zcela

intuitivní, ale též poskytuje dostatek informací o prováděných operacích a jejich vstupních parametrech. V úvodní části kapitoly došlo k záměrnému zopakování důležitých poznatků popsaných již v předchozích kapitolách. Důvodem byla skutečnost, že pojetí této kapitoly bylo uvažováno jako kompletní návod k použití programu a popsání jeho principů.

V této kapitole však nedošlo k popsání vlivu použité steganografie na nosiči a porovnání rozdílů původního obrazu a obrazu modifikovaného. Taktéž šlo o záměr, neboť předpokládám, že tato porovnání vlivu nastavení a použitého nosiče bude předmětem případných laboratorních úloh s programem pracujících, a do této kapitoly nebylo tedy zcela vhodné tato porovnávání a mé závěry uvádět. Těmto porovnáním se bude podrobně věnovat následující kapitola.

7 ANALÝZA POUŽITÉ STEGANOGRAFICKÉ METODY

Záměrem této kapitoly bude porovnávání vlivu použitého nosiče a to jak rozměrových vlastností, tak vhodností druhu obrazu uloženého v souboru. Druhem obrazu je myšleno, co daný obraz vyobrazuje (např.: krajina, portrét,…) a zdali je obraz barevný či černobílý. U černobílého obrazu se předpokládá uložení v TrueColor (24 bitů na pixel). A porovnání vlivu použité modifikační hloubky pro skrývání dat do obrazu.

7.1 STEGANOGRAFICKÁ KAPACITA V ZÁVISLOSTI NA VELIKOSTI OBRAZU NOSIČE

Steganografická kapacita je lineárně závislá na počtu pixelů v obraze, přičemž maximální steganografická kapacita je dána vztahem 7.1. Termínem maximální je míněn stav, kdy dochází k přepisu všech původních dat daty skrývanými (modifikační hloubka 8 bitů/bajt). To je však rozporuplné z titulu praktičnosti a vlastního záměru steganografie, což je přenos informace tak aby neznalý pozorovatel nebyl schopen probíhající komunikaci zpozorovat.

$$C = 3 \cdot n \tag{7.1}$$

kde C [bajt] – steganografická kapacita, n[-] – počet pixelů, 3 – konstanta udávající počet složek v pixelu.

Pro praktické využití bude tedy spíše žádanější steganografická kapacita závislá na použité modifikační hloubce. Ta je dána vztahem 4.1. Podle vztahu 4.1 je sestavena tabulka 7.1, ve které jsou pro nejzákladnější druhy rozlišení uvedeny steganografické kapacity v závislosti na modifikační hloubce.

Tab. 7.1: Závislost steganografické kapacity na velikosti nosiče a modifikační hloubce.

Vlastnostu nosiče			Modifikační hloubka [bit/bajt]							
Šířka	Výška	Rozlišení	1	2	3	4	5	6	7	8
x [px]	y[px]	Mpx[-]	C[MB]	C[MB]	C[MB]	C[MB]	C[MB]	C[MB]	C[MB]	C[MB]
320	240	0,077	0,028	0,056	0,084	0,113	0,141	0,169	0,197	0,225
640	480	0,307	0,113	0,225	0,338	0,450	0,563	0,675	0,788	0,900
800	600	0,480	0,176	0,352	0,527	0,703	0,879	1,055	1,230	1,406
1024	768	0,786	0,288	0,576	0,864	1,152	1,440	1,728	2,016	2,304
1152	864	0,995	0,365	0,729	1,094	1,458	1,823	2,187	2,552	2,916
1280	960	1,229	0,450	0,900	1,350	1,800	2,250	2,700	3,150	3,600
1400	1050	1,470	0,538	1,077	1,615	2,153	2,692	3,230	3,768	4,307
1600	1200	1,920	0,703	1,406	2,109	2,813	3,516	4,219	4,922	5,625
2048	1536	3,146	1,152	2,304	3,456	4,608	5,760	6,912	8,064	9,216
2800	2100	5,880	2,153	4,307	6,460	8,613	10,767	12,920	15,073	17,227
3200	2400	7,680	2,813	5,625	8,438	11,250	14,063	16,875	19,688	22,500
6400	4800	30,720	11,250	22,500	33,750	45,000	56,250	67,500	78,750	90,000

7.2 POROVNÁNÍ ROZDÍLŮ PO PROVEDENÉ MODIFIKACI U BAREVNÉHO A ČERNOBÍLÉHO OBRAZU

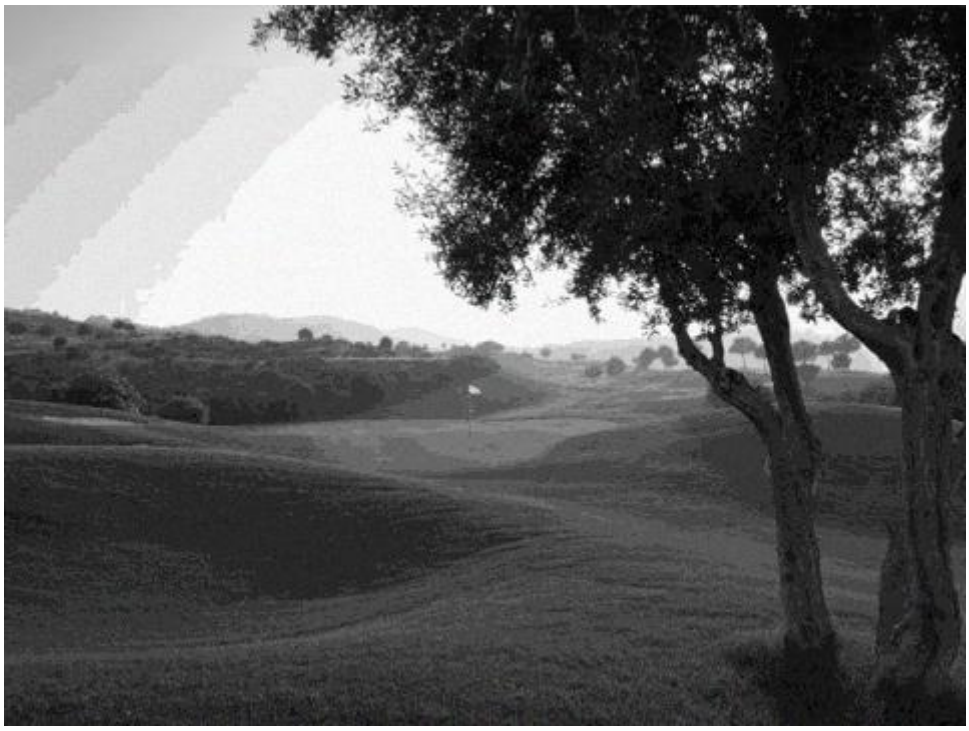
Experiment v této kapitole se bude zabývat porovnáním vhodnosti barevného nosiče vůči černobílému nosiči. Respektive u kterého obrazu bude menší pozorovatelný optický vliv po modifikaci obrazu.

V demonstrativním skrytí byla skrývána data o velikosti 230 kB do nosiče s rozlišením 480x360 px. Pro skrytí byla použita modifikační hloubka 4 bity/bajt. Barevný nosič se skrytými daty je zobrazen na obrázku 7.1 a černobílý nosič se skrytými daty je zobrazen na obrázku 7.2.

Při prvním zběžném porovnání je z obou obrazů patrné, že ke skrytí dat nebyly použity všechny pixely obrazu, což má za následek vytvoření viditelné hrany mezi modifikovanými a nemodifikovanými pixely. Tato hrana je nejlépe viditelná v levé horní části obou obrazů, kde je již pouze modrý odstín oblohy. Tento jev může způsobit prozrazení steganografické metody, protože pozorovatel má k dispozici i část původního obrazu a může provést porovnání. Způsob, kterým by se zamezilo vytvoření této hrany je popsán v kapitole 5.4. Další možností jak tento jev omezit by byla možnost modifikovaný obraz s vytvořenou hranou tzv. oříznout a to ze shora. Musel by však být kladen důraz na přesnost, kdy by nesmělo dojít k odstranění řádku pixelů s již modifikovanými pixely byť v minimálním množství. Odstranění třeba jen jednoho modifikovaného pixelu by zamezilo odkrytí skrytých dat.

Při dalším porovnávání obou obrazů je výsledkem zřejmý závěr a to ten, že ke skrývání dat do obrazu je vhodnějším nosičem barevný obraz. K tomuto závěru jsem došel zejména při porovnání viditelného poškození v oblasti trávníku, kde u černobílého nosiče je ve větší míře viditelný šum a vytvoření hran u přechodů jednotlivých odstínů. Kdežto u barevného nosiče je toto viditelné poškození menší a nejvíce pozorovatelné změny jsou pouze v oblasti stínů, kde též došlo k vytvoření hran mezi jednotlivými přechody odstínů. Důvod, proč jsou u černobílého obrazu změny viditelnější, než u barevného obrazu, je dle mého názoru evidentní. Lidské oko je nejvíce citlivé na jasové složky, dokáže tedy rozlišit menší změny v odstínu u jasových (černobílých) složek nežli u barevných složek. Na této rozdílné citlivosti lidského oka je též založeno nemálo kompresních metod obrazů, kde dochází k hrubšímu vzorkování barevných složek nežli jasových složek.

Obr. 7.1: Modifikovaný barevný nosič (480x360 px) s modifikační hloubkou 4 bity/bajt.



Obr. 7.2: Modifikovaný černobílý nosič (480x360 px) s modifikační hloubkou 4 bity/bajt.

7.3 VÝBĚR VHODNÉHO OBRAZU POUŽITÉHO JAKO NOSIČ

Výběrem správného obrazu použitého jako nosič pro skrývání dat, lze taktéž výrazně zvýšit bezpečnost použité steganografické metody. V následující části

provedu porovnání nosičů zachycující rozdílný typ obrazu. Pro samotnou demonstraci vhodného typu nosiče budou postačovat dvě názorné ukázky rozdílných typů nosičů, do kterých budou uložena stejná data se stejnou modifikační hloubkou, přičemž rozdíl bude patrný na první pohled.

a) původní nosič b) modifikovaný nosič

Obr. 7.3: Obrázek krajiny se skrytými daty s modifikační hloubkou 4 bity/bajt.

a) původní nosič b) modifikovaný nosič

Obr. 7.4: Obrázek oblohy se skrytými daty s modifikační hloubkou 4 bity/bajt.

Na obrázku 7.3 je zachycena různobarevná krajina s ostrými změnami barev a odstínů, oproti tomu je na obrázku 7.4 vyobrazena obloha s mírnými oblaky. Z porovnání obou typů obrázků lze stanovit jednoznačný závěr, že pro skrývání dat do obrazu je vhodnější použít pestré obrázky, ve kterých dochází k velkým kontrastům a změnám barev, ve kterých je snížení kvality méně pozorovatelné než na obrázku 7.4. Obrázek 7.4 je oproti předchozímu obrázku krajiny pouze v odstínech modré barvy a snížení kvality vlivem skrytí dat je zde již ve velké míře pozorovatelné. Ztrátou prvních 4 nejnižších bitů v každém bajtu má za následek menší rozlišení barevné hloubky. Tím dojde ke ztrátě relativně lineárních přechodů barvy, které se stanou změnami skokovými. V tomto důsledku dojde k vytvoření viditelných hran.

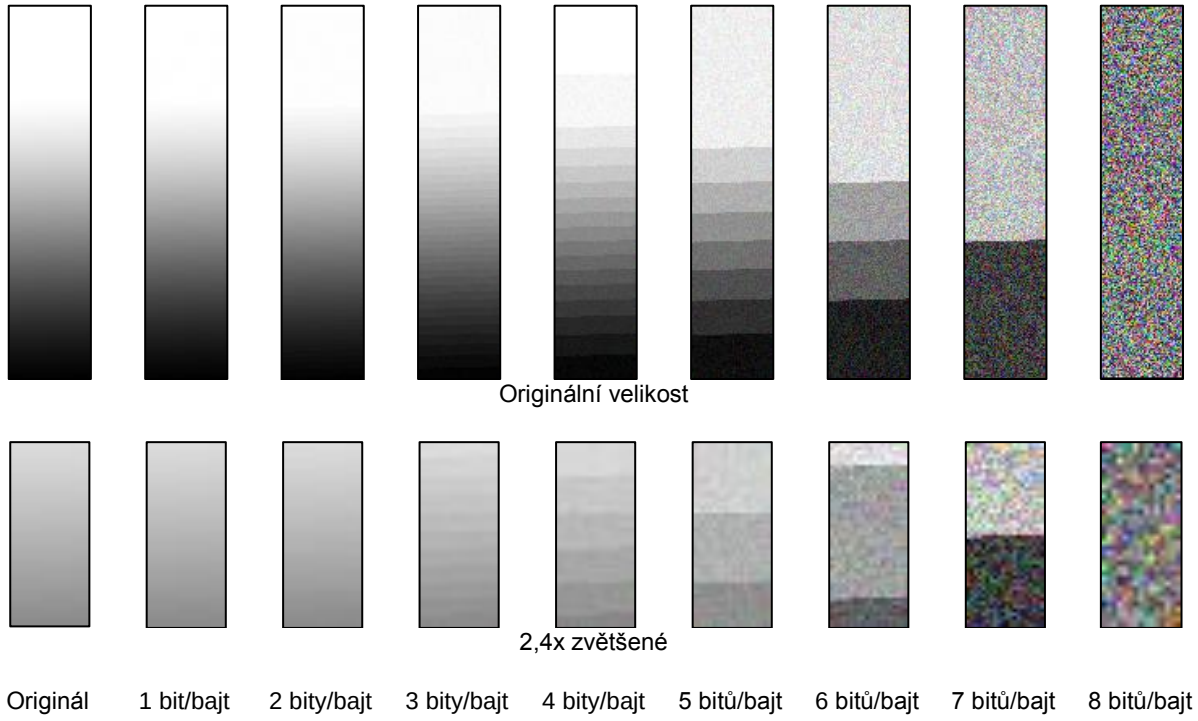
7.4 VLIV POUŽITÉ MODIFIKAČNÍ HLOUBKY NA KVALITU SKRYTÍ

Dle mého názoru má největší vliv na odolnost steganografické metody před prozrazením volba vhodné modifikační hloubky. Tato volba zásadně ovlivňuje výsledné poškození nosiče. Nedá se však obecně říci jaká maximální modifikační hloubka je optimální, každý obrazový soubor je jiný. Jak jsem demonstroval v předchozí kapitole (7.3) u některého nosiče při dané modifikační hloubce je výsledné poškození ještě v toleranci, kdežto u rozměrově totožného obrazu, který zobrazuje jinou situaci je při použití stejné modifikační hloubky poškození již vizuálně značné a mohlo by vést k prolomení steganografické metody.

Na následujících obrázcích 7.5 a 7.6 jsou provedena různá srovnání při volbě jednotlivých modifikačních hloubek. Na obrázku 7.5 je, jako nosič použit obrázek s bílým jednolitým pozadím kde je snadno pozorovatelný vzniklý šum a jeho rostoucí intenzita.

Obr. 7.5: Porovnání vzniklého šumu na nosiči při různých modifikačních hloubkách.

V následující sérii porovnání na obrázku 7.6 je jako nosič použit lineární černobílý přechod. Tento druh obrazu je dle mého názoru nejcitlivější na úbytek kvality vlivem skrytí obrazu. Snížením počtu bitů pro obraz se sníží množství reprodukovatelných barev, což má za následek tvrdší přechody mezi jednotlivými odstíny, což je zřejmé z obrázku 7.6.



Obr. 7.6: Porovnání poškození obrazu s lineárním přechodem pro různé modifikační hloubky.

Na obrázku 7.6 je patrné, zejména na zvětšeném detailu, že od modifikační hloubky 2 bity/bajt je již viditelná segmentace jednotlivých odstínů. K segmentaci dochází, jak již bylo dříve zmíněno, vlivem úbytku bitů nesoucích obrazová data a tím ke snížení počtu reprodukovatelných barev respektive odstínů. Původní obrazová data jsou zaznamenávána s barevnou hloubkou 8 bitů, počet reprodukovatelných odstínů je tedy 2^8 což je 256 úrovní v jedné složce. Počet reprodukovatelných úrovní jedné složky pro jednotlivé modifikační hloubky je uveden v tabulce 7.2.

Tab. 7.2: Vliv použité modifikační hloubky na počet úrovní barevné složky.

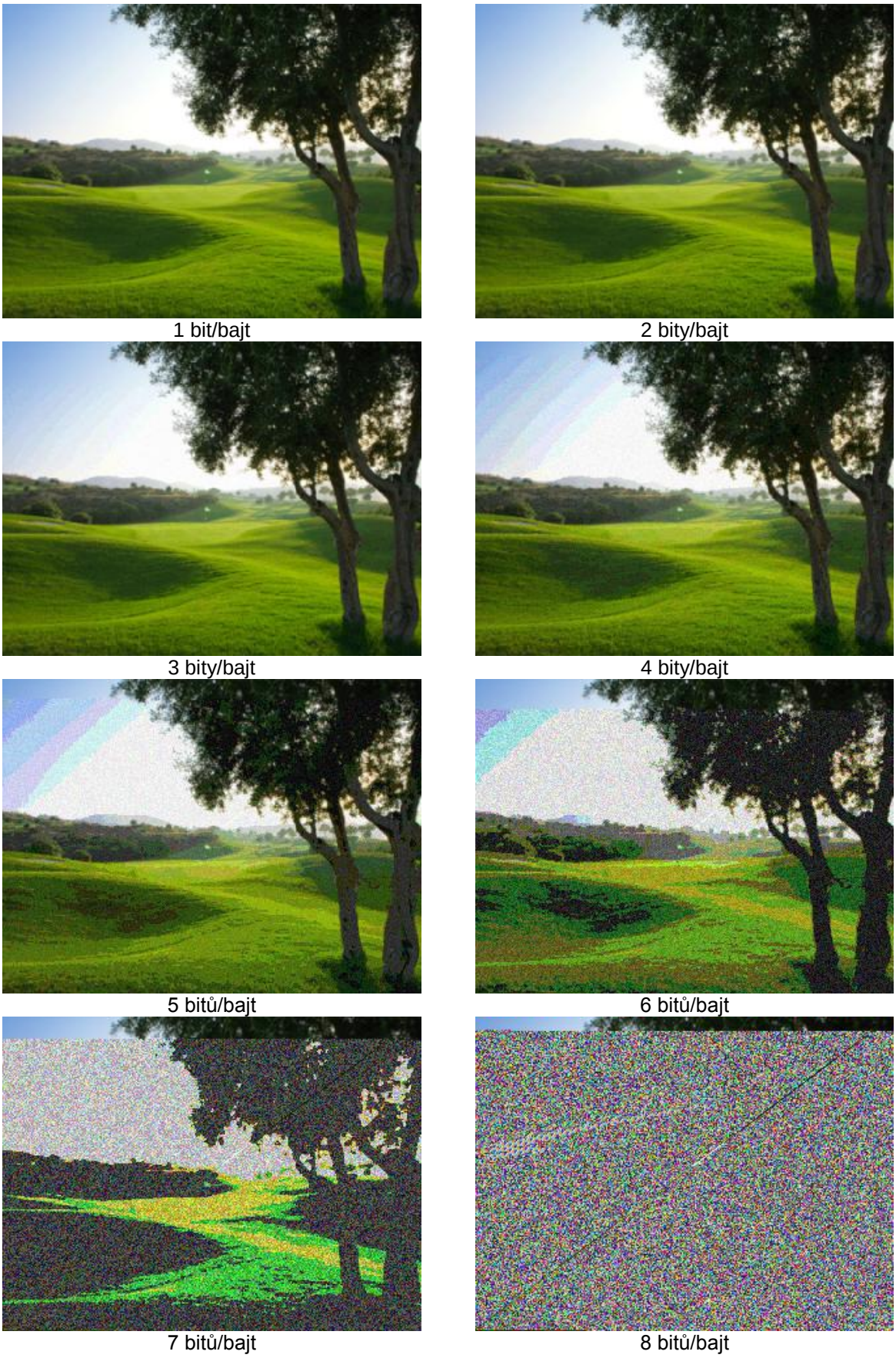
Modifikační hloubka	0	1	2	3	4	5	6	7	8
Obrazových bitů	8	7	6	5	4	3	2	1	0
Počet úrovní složky	**256**	**128**	**64**	**32**	**16**	**8**	**4**	**2**	**1**

Počet úrovní složky daný tabulkou 7.2 je též lehce spočítatelný i na obrázku 7.6, zejména ve vyšších úrovních modifikace, kde jsou vzniklé přechody snadno viditelné. Například na části, kde je zobrazen lineární přechod se skrytými daty s modifikační hloubkou 6 bitů/bajt, jsou zřejmé 4 segmenty.

Z tabulky 7.2 je viditelná exponenciální závislost počtu úrovní barevné složky z čehož lze vyvodit závěr, že se zvyšující se modifikační hloubkou dochází k lineárnímu růstu steganografické kapacity a exponenciálnímu poklesu kvality zobrazovaného obrazu.

V následující sérii reálných obrázků (obr. 7.8) je již prezentován vliv použité modifikační hloubky na nosič. Obrázek použitý jako nosič (obr. 7.7) je vybírán tak, aby obsahoval co možná nejvíce druhů zobrazení, jako jsou pozvolné přechody a prudké kontrasty. Na obrázku je fotografie golfového hřiště s částí oblohy a v popředí stojícími stromy. Přičemž obloha postupně přechází z modré až do téměř bílé barvy, trávník střídá kontrasty v zelené a obsahuje vržené stíny. Stromy v popředí mají tmavé kmeny a v koruně jsou veliké kontrasty mezi tmavým lupením a světlou oblohou.

Obr. 7.7: Originální obrázek použitý jako nosič.



Obr. 7.8: Série reálných obrázků s postupně rostoucí modifikační hloubkou.

Postupným porovnáním obrázků, na které byla použita pokaždé se zvyšující modifikační hloubka je závěr následující:

Pří modifikační hloubce 1 bit/bajt nelze pouhým okem pozorovat téměř žádné změny kvality nosiče. Zvýšením modifikační hloubky na 2 bity/bajt již došlo k mírné segmentaci v oblasti oblohy, ovšem daná segmentace je ještě zanedbatelná a hůře pozorovatelná. Ovšem u obrázku s modifikační hloubkou 3 bity/bajt, je již segmentace oblohy více než patrná. Tato modifikační hloubka by v tomto konkrétním případě již byla nevhodná z bezpečnosti steganografické metody. Ovšem, zanedbám-li segmentaci oblohy daný obraz je stále použitelný a na ostatních objektech jako jsou trávník a strom není snížení kvality obrazu stále pozorovatelné. V případě použití modifikační hloubky 4 bity/bajt je nejvyšší poškození opět v oblasti oblohy, přičemž zbytek obrazu je též ještě v použitelné toleranci. Modifikační hloubku 5 bitů/bajt je v tomto případě již zcela mimo použitelnou toleranci, obraz je zašumělý a segmentace se objevuje i v oblasti trávníku. Ostatní ukázky jsou již pouze spíše demonstrativního charakteru a v praxi nepoužitelné.

7.5 SHRNUTÍ

V této kapitole bylo záměrem poukázat na veškeré faktory ovlivňující bezpečnost steganografické metody, steganografickou kapacitu a na výběr vhodného nosiče. Jako stěžejní zjištění a důležitý fakt v této kapitole zmíněný považuji závislost použité modifikační hloubky na steganografickou kapacitu má lineární charakter ovšem na míru poškození obrazu má exponenciální charakter.

Dalším důležitým závěrem této kapitoly je skutečnost, že jednoznačně nelze určit obecnou doporučenou modifikační hloubku pro všechny nosiče, protože jak bylo v části 7.3 zmíněno, optimální modifikační hloubka závisí na obrazovém obsahu nosiče. Z části 7.4 lze však učinit závěr, že v praxi lze použít maximální modifikační hloubku 4 bity/bajt ve velice ojedinělých případech 5 bitů/bajt.

Největší omezení pro použití vyšších modifikačních hloubek jsou pozvolné přechody odstínu libovolné barvy, na kterých po provedení skrývání vzniknou hrany mezi odstíny.

Rovněž zajímavým zjištěním bylo, že pro skrývání dat jsou vhodnější barevné obrazy oproti černobílým obrazům, protože lidské oko je citlivější na jasové složky, kdežto na barevné je citlivé méně.

8 ZÁVĚR

Z obsahu této diplomové práce je patrné, že všeobecně je steganografie zajímavým oborem. Opakem steganografie je steganoanalýza, která se zabývá odhalováním skryté komunikace či ukrytých dat.

Použití tohoto typu steganografické metody, jaký je popsán v této diplomové práci, je založen na jednoduchém a snadno naprogramovatelném způsobu skrývání dat. Nedochází zde k žádné kompresi dat, což by danou metodu do jisté míry zkomplikovalo a též by vlivem komprese nejpravděpodobněji došlo ke snížení steganografické kapacity.

V této části bych také rád poukázal na skutečnost, že data skrytá v obraze BMP způsobem, který je popsán v této diplomové práci, nejsou odolná vůči poškození při jakékoli editaci obrazu, jako je upravení jasu, kontrastu, zmenšení, zvětšení, ořez, atd. Všechny tyto operace přenášená data poškodí, pouze v případě ořezu obrazu nemusí v případě popsaném v 7.2 dojít k poškození přenášených dat.

V závěrečné části byly navíc provedeny experimenty s vytvořeným programem při skrývání dat touto steganografickou metodou. Záměrem jednotlivých experimentů bylo poukázání na omezení dané steganografické metody, výběr vhodných nosičů a vlivu nastavení modifikační hloubky na viditelné poškození pro různé obrazy. Přičemž nejdůležitějšími závěry jsou skutečnosti, že nelze obecně doporučit optimální modifikační hloubku a skutečnost, že zvyšováním modifikační hloubky dochází k lineárnímu navyšování steganografické kapacity, ale zároveň k exponenciálně rostoucímu poškození obrazu použitého jako nosič.

Obr. 8.1: Informační dialogové okno programu *Data Camouflager*.

POUŽITÁ LITERATURA

[1] Žilka, R.: Steganografie a stegoanalýza. [Diplomová práce] Masarykova univerzita, Brno 2008.
[2] ČÍKA, P. Multimediální služby. Skriptum VUT v Brně. 2007. s. 1-106. ISBN: TKO 07-070.
[3] Tišnovský, P.: Grafický formát BMP - používaný a přitom neoblíbený. Root.cz, 19.10.2006. Dostupné na internetu: <http://www.root.cz/clanky/graficky-format-bmp-pouzivany-a-pritom-neoblibeny/>
[4] Bátora, J.: Bezpečné a spolehlivé ukrývání tajné zprávy ve zvukovém záznamu s podporou BCH kanálů a šifry AES. [Diplomová práce] Trenčínská univerzita Alexandra Dubčeka, Trenčín 2008.
[5] Vaněk, T.: Informační bezpečnost a utajování zpráv [3.přednáška - Moderní blokové šifry II], ČVUT, Praha 2009.
[6] Java ™ Cryptography Architecture (JCA) Reference Guide. Oracle.com, 2011. Dostupné na internetu <http://download.oracle.com/javase/6/docs/technotes/guides/security/crypto/CryptoSpec.html>
[7] Java SE, Oracle.com, 2011. Dostupné na internetu < http://www.oracle.com/technetwork/java/javase/>
[8] NetBeans IDE 7.0 Release Information, Netbeans.org, 2011. Dostupné na internetu <http://netbeans.org/community/releases/70/>

SEZNAM ZKRATEK, VELIČIN A SYMBOLŮ

AES	Šifrovací standart (Advanced Encryption Standard)
BMP	Bitmapový obrátek (BitMaP)
c [bit]	Steganografická kapacita v bitech
C [bajt]	Steganografická kapacita v bajtech
DEC	Desítková (decimální) číselná soustava
H [bit]	Modifikační hloubka
HEX	Šestnáctková (hexadecimální) číselná soustava
MF(jméno)	Vytvoření souboru daného jména (Make File)
Px	Pixel
RGB	Formát uložení obrazových souborů (Red, Green, Blue)
UI	Uživatelské rozhraní (User Interface)